LEVEL II

USAFA TR-79-5

AD A072527

FALCONFIX: A MULTI-MODAL APPROACH TO FIX COMPUTATION

WILLIAM T. HODSON III, LT COLONEL, USAF
JOSEPH C. H. SMITH, MAJOR, USAF

DEPARTMENT OF MATHEMATICAL SCIENCES
USAF ACADEMY, COLORADO 80840

JUNE 1979

FINAL REPORT

DDC FILE COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY

Editorial Review by Maj Ahern
Department of English
USAF Academy, Colorado 80840

This research report is presented as a competent treatment of the subject, worthy of publication. The United States Air Force Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the author.

This report has been cleared for open publication and/or public release by the appropriate Office of Information in accordance with AFR 190-17 and DODD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service.

This research report has been revised and is approved for publication.

John P. Wittry

JOHN P. WITTRY, Colonel, USAF
Vice Dean of the Faculty

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(17)

1. REPORT NUMBER: USAFA-TR-79-5
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

(6)

4. TITLE (and Subtitle): FALCONFIX: A Multi-Modal Approach to Fix Computation.

(9)

5. TYPE OF REPORT & PERIOD COVERED: Final Report.
6. PERFORMING ORG. REPORT NUMBER

(10)

7. AUTHOR(s): [redacted] William T. Hodson, III; [redacted] Joseph C.H. Smith
8. CONTRACT OR GRANT NUMBER(s): (12) 779

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Mathematical Sciences, United States Air Force Academy, CO 80840
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS: USAFA/DFMS, USAF Academy, CO 80840

(11)

12. REPORT DATE: June [redacted]979
13. NUMBER OF PAGES: 74

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

High-Frequency Direction-Finding (HFDF)
Fix Computation

79 08 9 013

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

FALCONFIX is a new algorithm for computing the location of an emitter given the bearings from a number of direction-finding stations located worldwide. This approach assumes that each reported bearing may be considered as a sample from a composite normal/uniform distribution, thereby explicitly acknowledging that the bearing may be either "true" or "wild." These distributions are used to create a likelihood function defined over the unit sphere. Non-linear programming techniques are then employed to identify all relative maxima (modes) of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

011 550

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (continued)

this function. The modes (which coincide with concentrations of bearings) are then ordered by value and reported as estimates of the emitter location.

ABSTRACT

FALCONFIX is a new algorithm for computing the location of an emitter given the bearings from a number of direction-finding stations located worldwide. This approach assumes that each reported bearing may be considered as a sample from a composite normal/uniform distribution, thereby explicitly acknowledging that the bearing may be either "true" or "wild." These distributions are used to create a likelihood function defined over the unit sphere. Non-linear programming techniques are then employed to identify all relative maxima (modes) of this function. The modes (which coincide with concentrations of bearings) are then ordered by value and reported as estimates of the emitter location.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the assistance rendered by LCDR Eric S. Hodson and CWO Anthony Untied of the Naval Security Group Command in the development of the FALCONFIX method. LCDR Hodson provided many valuable insights into the shortcomings of existing fix computation programs and consequently what areas should receive attention in the development of a new approach. CWO Untied translated the FORTRAN code into NELIAC for use on NSGC computers, thereby enabling the code to be tested against other programs using real operational data.

This work was supported in part by funding received from the Naval Security Group Command.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Background

A direction-finding system consists of both a net control station and a set of direction-finding sites at widely dispersed geographic locations. When instructed by the net control station, each site tunes its receiver to a specified frequency and, using extremely directional antennas, obtains a bearing on the signal it receives. These bearings are then reported to the net control station where they are processed to estimate the true location of the emitter (obtain a "fix").

Should all the bearings intersect at a single point, the estimate is, of course, obvious. Such a situation rarely occurs in practice since error is induced in the direction of the received signal by both the equipment at each site and by the propagation medium between the sites and the emitter. On the basis of empirical evidence, these errors appear to be approximately normally distributed. Thus, the reported bearings may be thought of as statistical samples taken from normal distributions with unknown true means. Various statistical techniques may be used to estimate these means, thereby providing an estimate of the true location of the emitter.

A particularly vexing aspect of the problem, however, is that of so-called "wild bearings." When a site tunes up on the specified frequency, it might not hear the same signal that other sites are receiving. This can happen both as a result of propagation conditions and of slight differences in the times the various sites listen. In the

latter case a site might report a bearing on the wrong emitter. From a statistical point of view, a wild bearing cannot properly be considered to be a sample from a normal distribution with mean bearing on the true emitter location. Unfortunately, it is impossible to distinguish between bearings taken on the correct emitter from bearings taken on another emitter prior to processing. All fix computation algorithms must deal with this problem in one way or another.

A common method is to consider the problem in two phases. In the first phase, the "most likely" set of true bearings is identified and in the second phase only these bearings are used in making the estimate. The difficulty with this approach is that in deciding which bearings are wild, it is necessary to establish a preliminary estimate of the true location. Then bearings which are not close to this position are considered wild and are rejected. If the preliminary estimate is not close to the true location, it is quite likely that good bearings will be thought to be wild and wild bearings good. It is also possible that enough bearings will be rejected so as to create a "no-fix" situation, i.e., not enough bearings remaining to reliably compute a fix.

Overview of the FALCONFIX Method

The method developed in this paper takes a considerably different approach to the problem--no bearing is ever rejected. Instead, the bearing reported from each site is considered to be a sample from a normal distribution with probability p (a good bearing) and from a uniform distribution with probability 1-p (a wild bearing), where p is determined empirically by observing the fraction of good bearings over a period of time. Then a likelihood function, the joint probability distribution of bearing errors assuming this composite normal/uniform distribution, is defined over a spherical earth model. The estimate of true emitter location is then taken as the location which maximizes the likelihood. Because of the nature of each of the bearing error distributions, this likelihood function may have numerous relative maxima (modes), corresponding to points on the earth where intersections of the reported bearings are concentrated.

Non-linear programming and regression techniques are then used to find the location and value of each mode of the likelihood function. The algorithm generates the three most likely modes and orders them according to the probability that they are associated with the true emitter location. The net control station operator may then determine which position to report, on the basis of both the algorithm's ordering as well as other available information.

Finally, the algorithm generates an elliptical confidence region with a specified probability of containing the true position of the emitter.

Chapter 2

MATHEMATICAL DESCRIPTION

Distribution of Bearing Error

The bearing which is reported from a direction-finding site may be taken on either the true target or, inadvertently, on a false target. If it is taken on a true target, we assume that the angular error (ε) between the reported bearing and the true bearing is a random variable with a normal distribution having mean zero and variance σ^2. On the other hand, if the bearing is taken on a false target, no information is conveyed by the reported bearing concerning the bearing to the true target. Therefore, we assume that the angular error is a random variable with a uniform distribution. Further, we assume that the bearing is taken on the true target with probability p and on the false target with probability (1-p); the value for p may be obtained from empirical evidence. For small values of ε, therefore, we assume the distribution to be normal and for larger values we assume it uniform. The transition points are designated $\pm c\sigma$, where c is a constant which depends upon both p and σ. Lastly, it is assumed that the probability density function of the bearing error is a continuous function of ε. The resulting distribution, which we shall call "composite normal-uniform" is represented by

$$h(\varepsilon)=\begin{cases} \dfrac{1}{K\sqrt{2\pi}\sigma} \exp(-\varepsilon^2/2\sigma^2), & -c\sigma\leq\varepsilon\leq c\sigma \\ \dfrac{1}{K\sqrt{2\pi}\sigma} \exp(-c^2/2), & -\pi\leq\varepsilon<-c\sigma \text{ or } c\sigma<\varepsilon\leq\pi \end{cases}$$

where ϵ and σ are in radians and the normalizing constant K is given by

$$K = [2\Phi(c)-1] + 2[\pi-c\sigma]\left[\frac{\exp(-c^2/2)}{\sqrt{2\pi}\sigma}\right]$$

where Φ is the standard normal cumulative distribution function.

The graph of a typical composite normal-uniform is depicted in Figure 1 below. The fact that the ordinate is constant for values of ϵ greater than $c\sigma$ or less than $-c\sigma$ is extremely significant to the method, as shall become apparent in later sections.

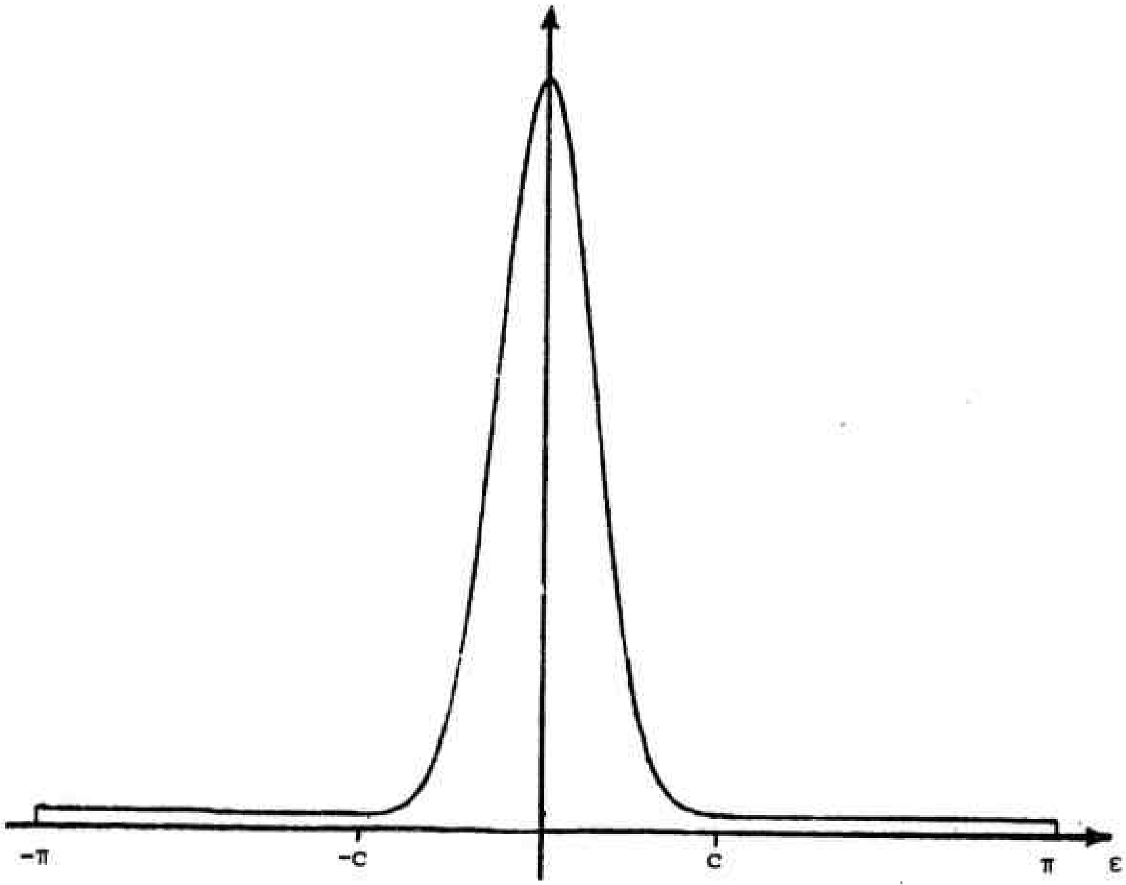


Figure 1. Typical Composite Normal-Uniform PDF

To determine the transition point, we note that

$$p = \frac{1}{K}\int_{-c\sigma}^{c\sigma} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\varepsilon^2/2\sigma^2)\,d\varepsilon$$

$$= \frac{1}{K}\,[2\Phi(c)-1], \text{ or}$$

$$p = \frac{2\Phi(c)-1}{[2\Phi(c)-1] + 2[\pi-c\sigma]\left[\frac{\exp(-c^2/2)}{\sqrt{2\pi}\sigma}\right]}$$

In Figure 2 below, this relationship between p and σ is plotted for various values of c. Then, given values of p and σ, c can be obtained by interpolating between the curves.

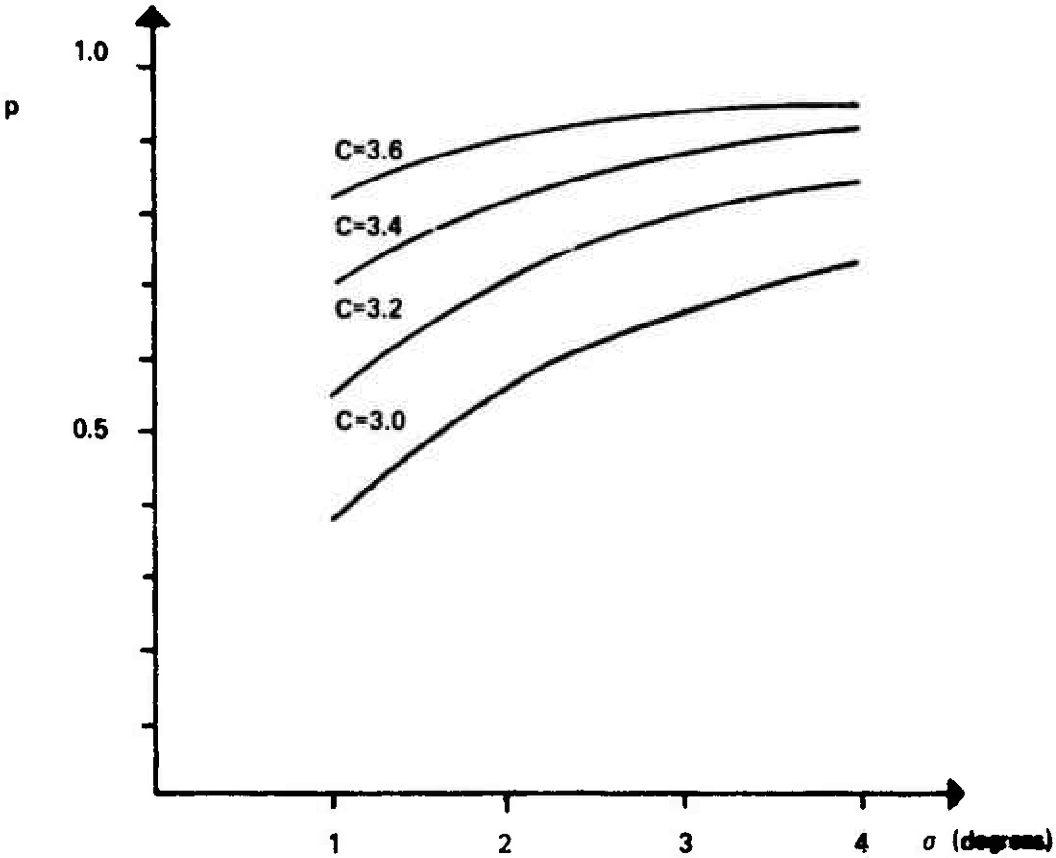


Figure 2. Transition Point Curves

One final approximation for the density function is required to make the procedure more efficient computationally. The ε^2 term is difficult to work with using vector methods, so the Maclaurin approximation for cos ε is used to derive the relationship

$$\varepsilon^2 \approx 2(1 - \cos \varepsilon).$$

This approximation is correct to within 0.4% for values of ε up to 0.2 radians, a value greater than will occur in practice.

Thus, $h(\varepsilon)$ is approximated by

$$f(\varepsilon) = \begin{cases} \dfrac{1}{K\sqrt{2\pi}\sigma} \exp\left[\dfrac{-1}{\sigma^2}(1-\cos \varepsilon)\right], & -c\sigma \le \varepsilon \le c\sigma \\ \dfrac{1}{K\sqrt{2\pi}\sigma} \exp\left[\dfrac{-1}{\sigma^2}(1-\cos(c\sigma))\right], & -\pi \le \varepsilon < -c\sigma \text{ or } c\sigma < \varepsilon \le \pi \end{cases}$$

The Likelihood Function

The likelihood function for the emitter location (ϕ,θ) is defined to be:

$$F(\phi,\theta) = G(\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_n) = f_1(\varepsilon_1)f_2(\varepsilon_2)\cdots f_n(\varepsilon_n)$$

where ε_i is the angular error between the reported bearing from station i and the true bearing from station i to (ϕ,θ). Thus ε_i is a random variable with density function

$$f_i(\varepsilon_i) = \begin{cases} \dfrac{1}{K_i\sqrt{2\pi}\sigma_i} \exp\left[-\dfrac{1}{\sigma_i^2}(1-\cos\varepsilon_i)\right], & -c_i\sigma_i \le \varepsilon_i \le c_i\sigma_i \\ \dfrac{1}{K_i\sqrt{2\pi}\sigma_i} \exp\left[-\dfrac{1}{\sigma_i^2}(1-\cos(c_i\sigma_i))\right], & -\pi \le \varepsilon_i < -c_i\sigma_i \text{ or } c_i\sigma_i < \varepsilon_i \le \pi \end{cases}$$

Therefore,

$$F(\phi,\theta) = \frac{1}{(2\pi)^{n/2}} \prod_{i=1}^{n} \frac{1}{K_i\sigma_i} \prod_{i\in W} \exp\left(\frac{-1}{\sigma_i^2}(1-\cos\varepsilon_i)\right) X \prod_{i\notin W} \exp\left(\frac{-1}{\sigma_i^2}(1-\cos(c_i\sigma_i))\right)$$

where $W = \{i : |\varepsilon_i| \le c_i\sigma_i\}$

As will be made clear subsequently, our approach to estimating emitter location will be to find the relative maxima--which we call "modes"--of this function.

Since $\dfrac{1}{(2\pi)^{n/2}} \prod_{i=1}^{n} K_i\sigma_i$ is not a function of the ε_i, $F(\phi,\theta)$ is maximized whenever

$$F^*(\phi,\theta) = \prod_{i\varepsilon W} \exp\left(-\frac{1}{\sigma_i^2}(1-\cos\,\varepsilon_i)\right) \prod_{i\notin W} \exp\left(-\frac{1}{\sigma_i^2}(1-\cos(c_i\sigma_i))\right)$$

is a maximum. $F^*(\phi,\theta)$, in turn, is maximized whenever its logarithm

$$L(\phi,\theta) = \sum_{i\varepsilon W} \frac{-1}{\sigma_i^2}(1-\cos\,\varepsilon_i) + \sum_{i\notin W} \frac{-1}{\sigma_i^2}(1-\cos(c_i\sigma_i))$$ is, and this is the

function to which we shall direct our attention.

Before proceeding with the development of the method, we shall pause to consider exactly what this function $L(\phi,\theta)$ looks like. In Figures 3 through 7 are depicted five examples of a set of bearings and the resulting function L.

In the figure below the case in which two lines of bearing intersect at a single point is illustrated. Notice that the function has a single mode at the point of intersection. (As we shall see later, the FALCONFIX algorithm does not consider two bearing crosses as potential position extimates.)

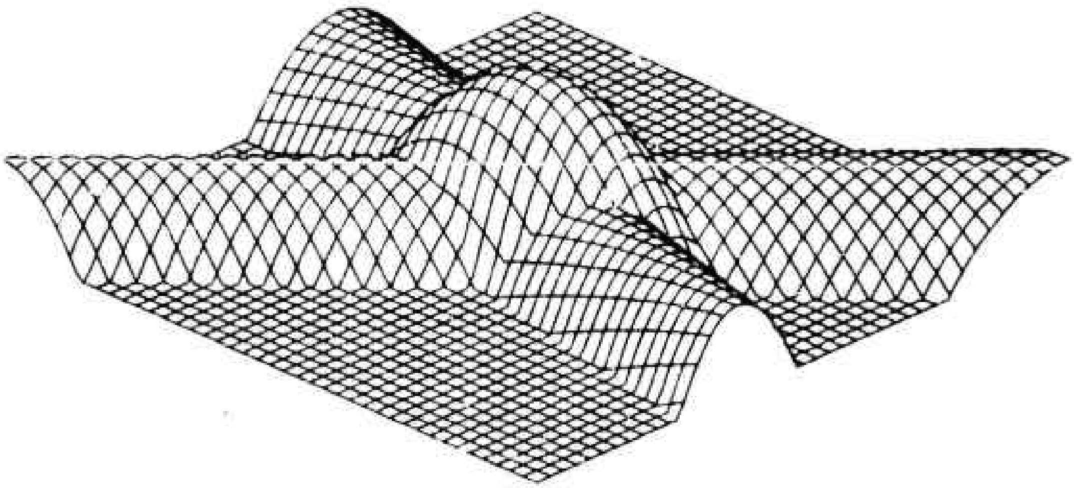


Figure 3. $L(\phi,\theta)$; Two Bearing Cross

In Figure 4 below, the function $L(\phi,\theta)$ in which three bearings intersect at a point is illustrated. Here the mode is somewhat more sharply peaked than in the previous case.

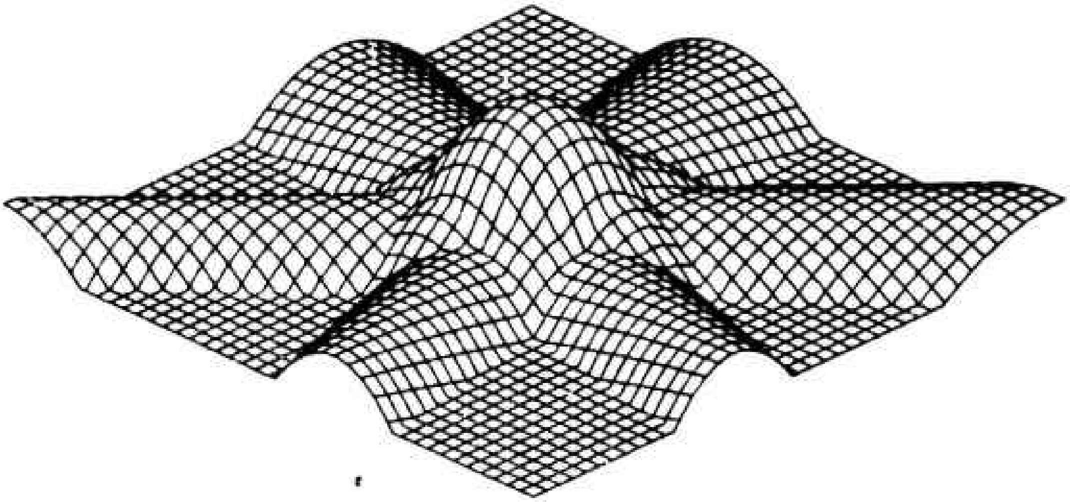


Figure 4. $L(\phi,\theta)$; Three Bearing Case #1

In Figure 5, the three bearings do not meet at a single point, and, in fact, the "cocked hat" is so large compared to the standard deviation of the bearing errors, that the function does not have a single mode in the center. Instead, there are three individual two bearing crosses at each of the intersections.

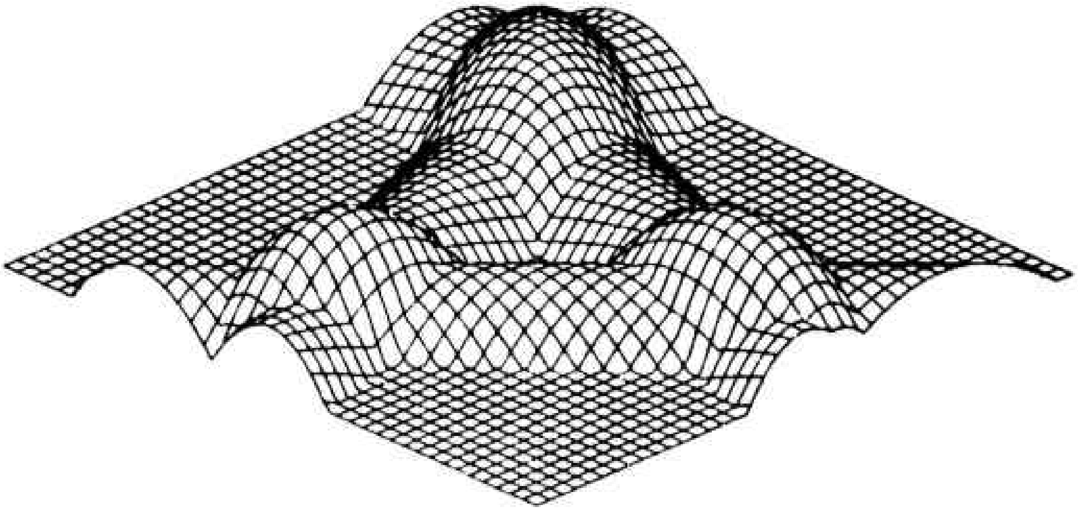


Figure 5. $L(\phi,\theta)$; Three Bearing Case #2

In Figure 6 below, a four bearing case is depicted in which the bearings do not intersect at a single point. Notice that this figure is considerably less regular than the previous figures, and is more typical of what one finds in real cases.

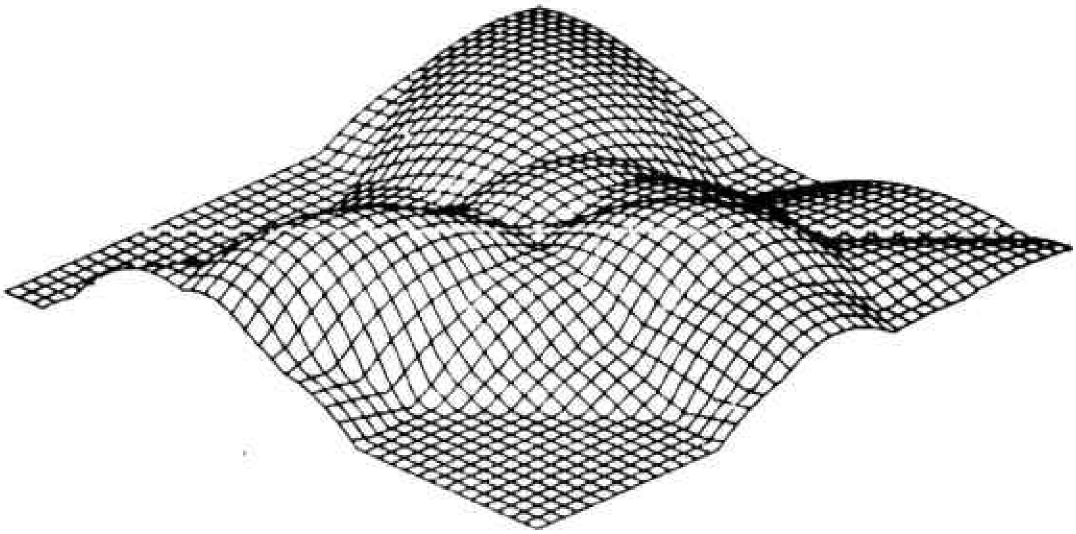


Figure 6. $L(\phi,\theta)$; Four Bearing Case

In Figure 7 the bimodal case is illustrated. In this five bearing example it is not likely that all bearings were taken on the same emitter. Rather, it is much more plausible that either Bearings 1, 2 and 3 are good bearings (with 4 and 5 wild) or Bearings 1, 4 and 5 are good bearings (with 2 and 3 wild). The two modes in the figure correspond to these two possibilities.

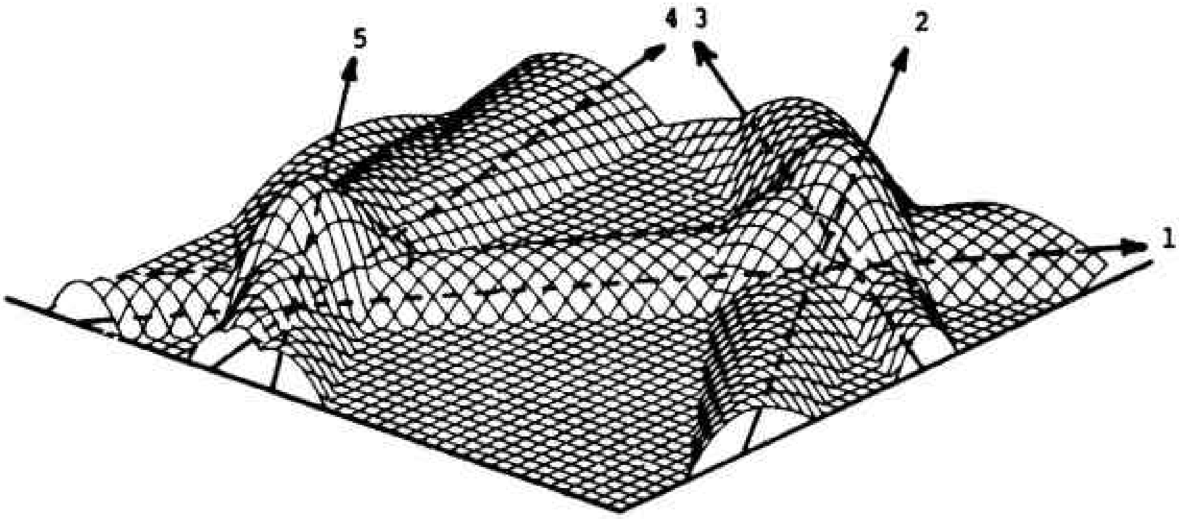


Figure 7. $L(\phi,\theta)$; Bimodal Case

Before going on to the methodology of identifying these modes, it will first be necessary to express each of the ε_i in the equation for L explicitly as functions of ϕ and θ, the latitude and longitude of the position estimate. To accomplish this we will rely heavily on the use of vectors as described in the next section.

Vector Representation

A position vector P from the center of the earth to a point (ϕ,θ) on the surface may be represented as

$$P = (\cos\theta \cos\phi)\bar{i} + (\sin\theta \cos\phi)\bar{j} + (\sin\phi)\bar{k}$$

This vector P is depicted in Figure 8 below in a right-hand coordinate system in which $\bar{i}$ points to the prime meridian on the equator and $\bar{k}$ points to the north pole. Thus, east longitude is represented by positive values of θ, west longitude by negative values, north latitude by positive values of ϕ and south latitude by negative values.

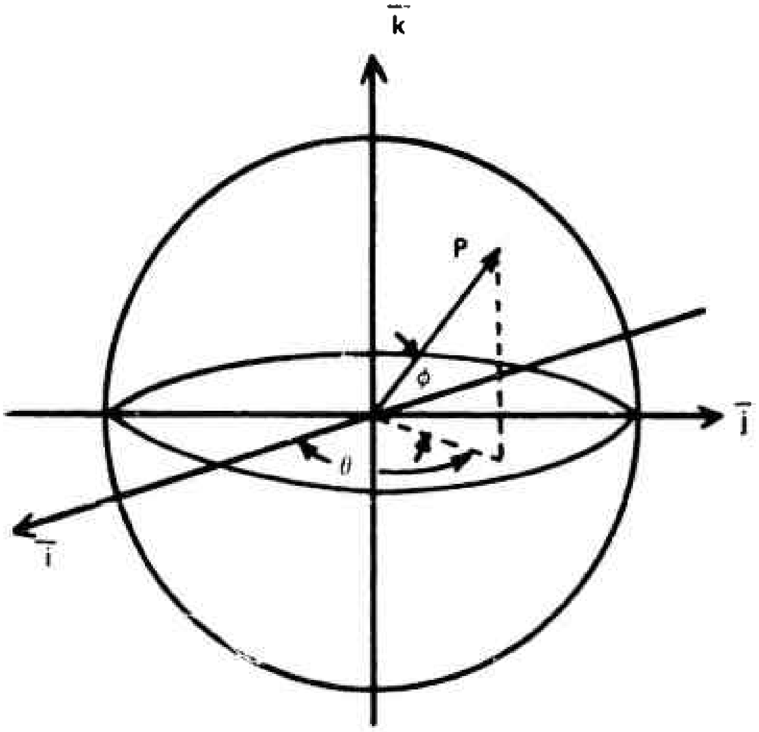


Figure 8. Position Vector

If (ϕ,θ) is the location of the target and (ϕ_i,θ_i) is the location of the i^{th} station, then the corresponding position vectors are:

$$T = (\cos\theta \cos\phi)\bar{i} + (\sin\theta \cos\phi)\bar{j} + (\sin\phi)\bar{k}$$
$$S_i = (\cos\theta_i \cos\phi_i)\bar{i} + (\sin\theta_i \cos\phi_i)\bar{j} + (\sin\phi_i)\bar{k}$$

Let β_i be the bearing reported by the i^{th} station. Then in terms of the local coordinate system $(\bar{i}^*, \bar{j}^*, \bar{k}^*)$ in which $\bar{i}^*$ points east, $\bar{j}^*$ points north and $\bar{k}^*$ points to the local zenith, a vector pointing in the direction of β_i may be represented as

$$B_i^* = (\sin\beta_i)\bar{i}^* + (\cos\beta_i)\bar{j}^*$$

which is shown in the figure below.

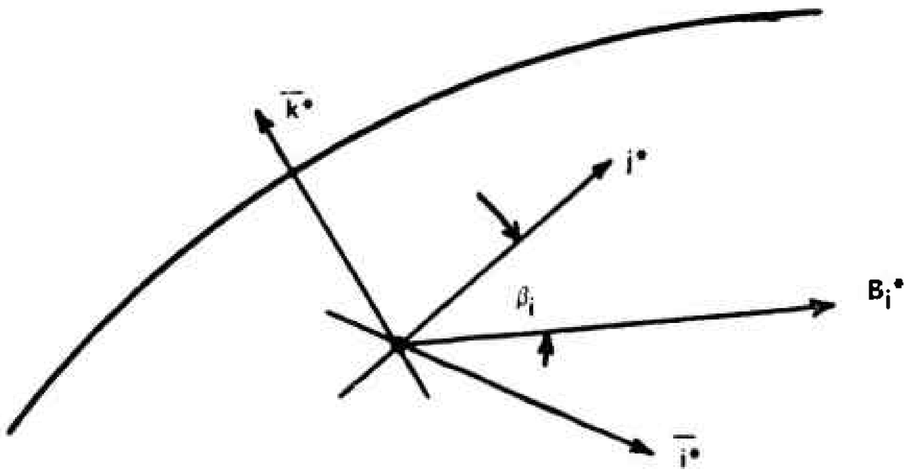


Figure 9. Bearing Vector

To be useful, however, it is necessary to express each bearing vector in the same coordinate system as the T and S_i vectors. This is accomplished by rotating the local frame with transformation R_i so that it coincides with the geocentric frame shown in Figure 8.

Specifically,

$$B_i = R_i B_i^* = A\bar{i} + B\bar{j} + C\bar{k} \quad \text{where}$$

$$A = -(\sin\phi_i \cos\theta_i \cos\beta_i + \sin\theta_i \sin\beta_i)$$

$$B = \cos\theta_i \sin\beta_i - \sin\phi_i \sin\theta_i \cos\beta_i$$

$$C = \cos\phi_i \cos\beta_i$$

Now, T, the S_i and the B_i are all unit vectors expressed in the same geometric frame.

Then, the vector $\left(\frac{S_i \times T}{|S_i \times T|}\right)$ is a unit vector normal to the "station-target plane," the plane containing S_i and T. Similarly, $\left(\frac{S_i \times B_i}{|S_i \times B_i|}\right)$ is a unit vector normal to the "station-bearing plane," the plane containing S_i and B_i.

Clearly, the angle between these two normal vectors is equal to the dihedral angle between the two planes, which in turn is equal to the angle ε_i between the reported bearing and the true bearing from the i^{th} station to the target. This relationship is shown in Figure 10.

From the familiar dot product relationship we have

$$\cos\varepsilon_i = \left(\frac{S_i \times T}{|S_i \times T|}\right) \cdot \left(\frac{S_i \times B_i}{|S_i \times B_i|}\right)$$

Since S_i and B_i are orthonormal, it is true that $|S_i \times B_i| = 1$, which means that

$$\cos\varepsilon_i = \frac{(S_i \times T) \cdot (S_i \times B)}{|S_i \times T|}$$

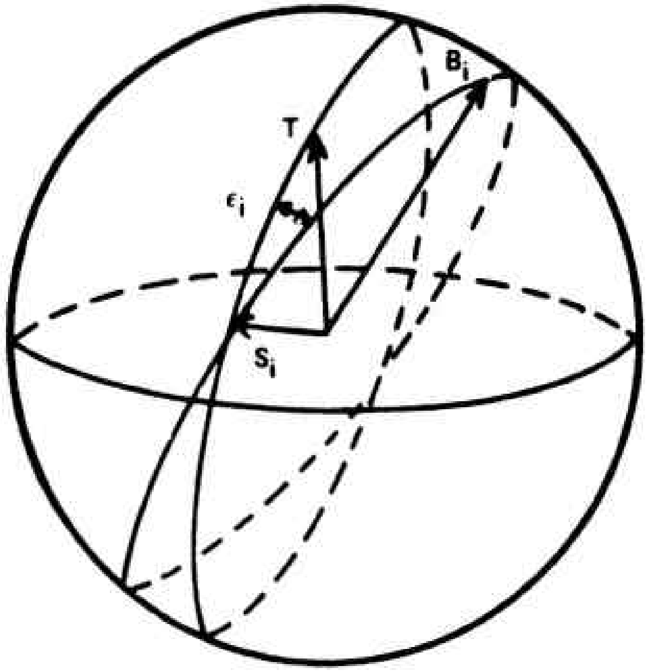


Figure 10. Dihedral Angle

Employing a little-used identity from vector calculus on the numerator of this expression yields

$$(S_i \times T)\cdot(S_i \times B_i) = (B_i \cdot T)(S_i \cdot S_i) - (S_i \cdot T)(S_i \cdot B)$$

But $S_i \cdot S_i = 1$ since S_i is a unit vector and $S_i \cdot B_i = 0$ since S_i and B_i are orthogonal.

Thus $(S_i \times T)\cdot(S_i \times B_i) = B_i \cdot T$, which means that

$$\cos \varepsilon_i = \frac{B_i \cdot T}{|S_i \times T|}$$

Now $|S_i \times T|$ is equal to the sine of the angle between S_i and T. Since $S_i \cdot T$ equals the cosine of the same angle, we can represent $|S_i \times T|$ as

$$|S_i x T| = \sqrt{1-(S_i \cdot T)^2}$$

which is somewhat easier to deal with computationally. The relationship

$$\cos \varepsilon_i = \frac{B_i \cdot T}{\sqrt{1-(S_i \cdot T)^2}}$$

will be used in subsequent calculations. Making this substitution, the logarithm of the likelihood function becomes:

$$L(\phi,\theta) = \sum_{i \varepsilon W} \frac{-1}{\sigma_i^2} \left(1 - \frac{B_i \cdot T}{\sqrt{1-(S_i \cdot T)^2}}\right) + \sum_{i \notin W} \frac{-1}{\sigma_i^2} (1-\cos(c_i \sigma_i))$$

In this equation the c_i, σ_i, B_i and S_i are all constants and T is a function of ϕ and θ. We now have $L(\phi,\theta)$ in a form where it is possible to find the modes by moving T in such a way as to progressively increase the value of L. In the next section we will illustrate exactly how this is done.

Modes of $L(\phi,\theta)$

$L(\phi,\theta)$ can be written in the form:

$$L(\phi,\theta) = \sum_{i\varepsilon W} \frac{B_i\cdot T}{\sigma_i^{2}\sqrt{1-(S_i\cdot T)^2}} + K$$

where K = a constant

$$B_i = b_{i1}\bar{i} + b_{i2}\bar{j} + b_{i3}\bar{k}$$

$$S_i = s_{i1}\bar{i} + s_{i2}\bar{j} + s_{i3}\bar{k}$$

$$T = t_1\bar{i} + t_2\bar{j} + t_3\bar{k}$$

$$= \cos\theta\cos\phi\ \bar{i} + \sin\theta\cos\phi\ \bar{j} + \sin\phi\ \bar{k}$$

and $|B_i| = |S_i| = |T| = 1$

A necessary condition for the point (ϕ^*,θ^*) to be a mode (relative maximum) of $L(\phi,\theta)$ is that both

$$\left.\frac{\partial L}{\partial\theta}\right|_{\theta=\theta^*} = 0 \quad \text{and} \quad \left.\frac{\partial L}{\partial\phi}\right|_{\phi=\phi^*} = 0$$

After considerable calculation it can be shown that:

$$\frac{\partial L}{\partial\theta} = \sum_{i\varepsilon W} F_i(T)\left(\frac{-b_{i1}\sin\theta\cos\phi + b_{i2}\cos\theta\cos\phi}{B_i\cdot T} + \frac{(S_i\cdot T)(-s_{i1}\sin\theta\cos\phi + s_{i2}\cos\theta\cos\phi)}{1-(S_i\cdot T)^2}\right), \text{ and,}$$

$$\frac{\partial L}{\partial\phi} = \sum_{i\varepsilon W} F_i(T)\left(\frac{-b_{i1}\cos\theta\sin\phi - b_{i2}\sin\theta\sin\phi + b_{i3}\cos\phi}{B_i\cdot T} + \frac{(S_i\cdot T)(-s_{i1}\cos\theta\sin\phi - s_{i2}\sin\theta\sin\phi + s_{i3}\cos\phi)}{1-(S_i\cdot T)^2}\right)$$

$$\text{where } F_i(T) = \frac{B_i \cdot T}{\sigma_i^2 \sqrt{1-(S_i \cdot T)^2}}$$

Setting these partial derivatives equal to zero and then solving for θ and ϕ respectively, we obtain:

$$\theta = J(\phi,\theta) = \tan^{-1}\left\{\frac{\sum_{i\varepsilon W} F_i(T)\left(\frac{b_{i2}}{B_i \cdot T} + \frac{(S_i \cdot T)s_{i2}}{1-(S_i \cdot T)^2}\right)}{\sum_{i\varepsilon W} F_i(T)\left(\frac{b_{i1}}{B_i \cdot T} + \frac{(S_i \cdot T)s_{i1}}{1-(S_i \cdot T)^2}\right)}\right\}$$

and

$$\phi = K(\phi,\theta) = \tan^{-1}\left\{\frac{\sum_{i\varepsilon W} F_i(T)\left(\frac{b_{i3}}{B_i \cdot T} + \frac{(S_i \cdot T)s_{i3}}{1-(S_i \cdot T)^2}\right)}{\sum_{i\varepsilon W} F_i(T)\left(\frac{b_{i1}\cos\theta + b_{i2}\sin\theta}{B_i \cdot T} + \frac{(S_i \cdot T)(s_{i1}\cos\theta + s_{i2}\sin\theta)}{1-(S_i \cdot T)^2}\right)}\right\}$$

Thus, at the point (ϕ^*,θ^*) these equations will be satisfied. These equations are used to establish the following recursion relationships:

$$\theta_{k+1} = J(\phi_k,\theta_k) \quad \text{and} \quad \phi_{k+1} = K(\phi_k,\theta_{k+1}),\ k = 0,1,2,\ldots$$

The FALCONFIX algorithm for estimating (ϕ^*,θ^*) proceeds as follows:

1. Select a starting point (ϕ_0,θ_0). (The selection procedure is discussed later.)

2. $T_0 = T(\phi_0,\theta_0)$: Initialize the target vector

3. $k = 0$: Initialize k

4. $\delta = \delta_1$: Specify the stopping rule

5. $\theta_{k+1} = J(\phi_k, \theta_k)$: Calculate the next longitude

6. $\phi_{k+1} = K(\phi_k, \theta_{k+1})$: Calculate the next latitude

7. $T_{k+1} = T(\phi_{k+1}, \theta_{k+1})$: Calculate the next target vector

8. If $k>M$ then $\delta=\delta_2$: Relax, the stopping rule if k is large

9. If $|T_{k+1}-T_k|<\delta$ or $k>N$ then go to 12 : stopping rule

10. $k = k+1$: Increment k

11. Go to 5 : Perform the next iteration

12. $(\phi_{k+1}, \theta_{k+1})$ is an estimate for (ϕ^*, θ^*)

Note: $N > M$ and $\delta_2 > \delta_1$

The intersections of pairs of bearings are used as the starting points for the algorithm. Those pairs which intersect at an angle of greater than 20° are considered first—those intersecting at smaller angles are deferred for later consideration. The first check in determining whether an intersection constitutes a valid starting point is if that particular pair is not a subset of a bearing set W from a previously generated mode. If it is, the intersection is rejected since the iteration procedure would yield the same mode. The second check is to determine if at least one additional bearing is contained in the set W at the point of intersection. If not, the intersection is also rejected as a starting point since in this case no iteration will take place and two-bearing modes are not allowable, except in the special case where only two bearings are input.

The following example illustrates the algorithm and the selection of starting points. Suppose the reported bearings are as shown in

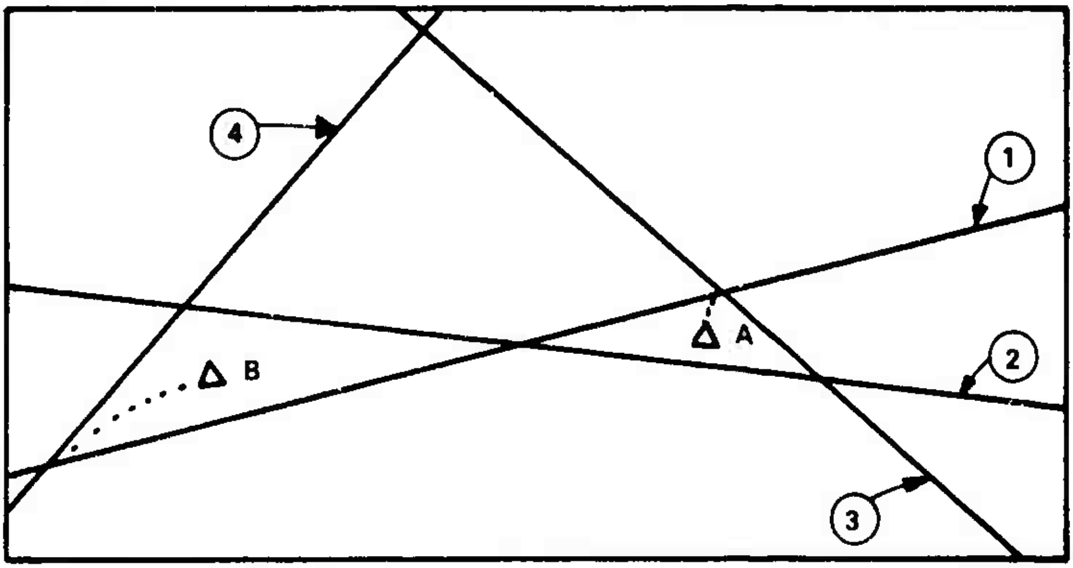


Figure 11. Example of Mode Determination

Figure 11. The intersection (1,2) is temporarily set aside because the angle is too small. (1,3) satisfies this criterion and also that at least one additional bearing, (2), is contained in the bearing set at this point. Therefore (1,3) is a valid starting point. The algorithm commences and after several iterations converges to the point A. Since bearings 1, 2 and 3 are in this bearing set, the intersections (1,2), (1,3) and (2,3) are removed from further considerations. (1,4) is the next candidate and since it meets both criteria it is selected as a starting point. The algorithm begins there and converges to point B. In the same way as before, (1,2), (1,4) and (2,4) are eliminated. (3,4) is the only intersection remaining. It satisfies the angular

criterion, but no other bearing is close enough to be a member of the bearing set at the point of intersection. Therefore (3,4) is eliminated. Since no intersections remain as candidates, all modes have been found and the problem is terminated.

When the algorithm generates more than a single mode, we order them with respect to the probability that they coincide with the true location of the emitter. Since each bearing has a probability p of being a true bearing and for realistic direction-finding systems, this value is greater than 0.5, the mode having the greatest number of forward bearings in its bearing set is considered the most likely. In the case where two or more modes involve the same number of forward bearings, the algorithm uses the degree of dispersion of the bearings as a secondary criterion. Specifically, the average squared perpendicular distance from the position estimate to each line of bearing is computed and the modes are ordered so that the one with the smallest such value is first.

Standard Deviation of Bearing Error

The standard deviation of the bearing error (σ) is used in the calculation of the likelihood function and its partial derivatives. While the particular method used for obtaining σ is not crucial to the FALCONFIX algorithm, for the sake of completeness its computation will be discussed.

The standard deviation is taken to be a function of both the characteristics of the particular receiving site (antenna, equipment, location, etc.) and of the distance between the site and the emitter location. In particular, we assume the multiplicative relationship $\sigma = h(d)\bar{\sigma}$ where $\bar{\sigma}$ is a constant representing the normalized standard deviation of the bearing error from the site and $h(d)$ is an adjustment for distance.

If $s = \sin \delta$, where δ is the central angle between the target vector T and the station vector S, then

$$d = 3444 \sin^{-1}(s) \quad , d \text{ in nautical miles}$$

or

$$s = \sin\left(\frac{d}{3444}\right)$$

The function $h(d)$ below is motivated by the Ross Range Curve, although certain modifications have been made.

$$h(d) = \begin{cases} 60 & , \text{ if } d \leq 60 \\ 60-3152.387(s-.0174524) & , \text{ if } 60 < d \leq 120 \\ .174524/s & , \text{ if } 120 < d \leq 600 \\ s + .826352 & , \text{ if } 600 \leq d \leq 5410 \\ 1.826352 & , \text{ if } d > 5410 \end{cases}$$

A graph of $h(d)$ is provided in Figure 12 below.

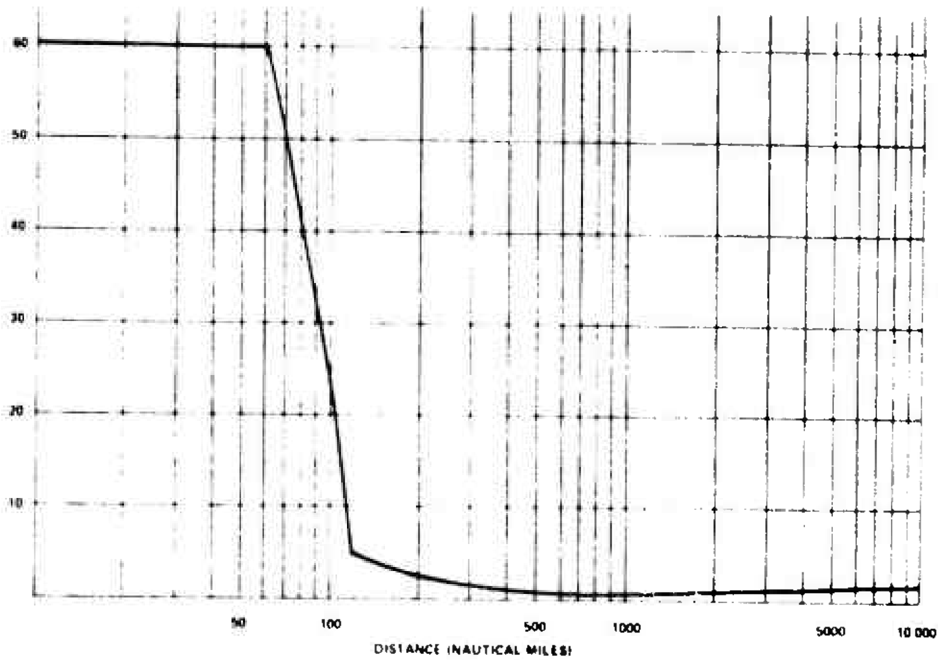


Figure 12. Graph of h(d)

The extremely large standard deviation for distances less than 100 nautical miles or so, is essentially an artifice to assure inclusion of bearings from sites which are close to a potential mode; such bearings are almost invariably good bearings. For example, if $\bar{\sigma}$ is equal to 1.0 degrees and if the transition point c (referred to in Figure 2) is set equal to 3.0, then any estimate of emitter location within the "keyhole" in Figure 13 will include the bearing from site s. (In this figure the origin represents the site and the horizontal axis is the direction of the reported bearing). Notice that the bearing is included whenever the site is within 60 nautical miles of the estimate.

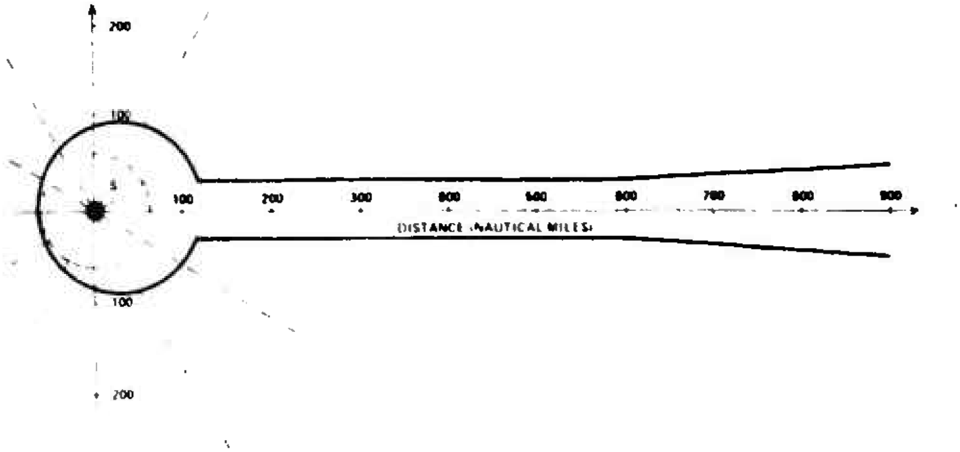


Figure 13. Inclusion Region

The REFIX Procedures

One final procedure is used to refine the position estimates generated by the previously described algorithm. The need for such a termination procedure is two-fold.

- If one of more of the bearings in the included set is from a station which is reasonably close to the position estimate, the artificially large standard deviation associated with such bearings will tend to diminish its rightful impact on the position estimate.
- If the likelihood function happens to be unusually flat in the neighborhood of the mode, it is possible that the stopping rule may terminate the algorithm prematurely, resulting in a reported position estimate which is not coincident with the point at which the relative maximum occurs.

Two different termination procedures have been developed, but both have the following features in common.

- The bearing set is "frozen" to include only those bearings which were included in the final iteration of the algorithm previously described.
- The standard deviations of bearings from stations within 600 nautical miles of the position are modified by changing $h(d)$ to $h^*(d) = 0.75 + d/2400$ where d is the distance in nautical miles.

REFIX/1. In this procedure, after the bearing set has been frozen and the standard deviations of bearings from close-in stations have been modified, the iterative algorithm is employed again.

REFIX/2. This termination procedure uses techniques employed in multiple regression analysis. Most of the computation done here is required for the confidence region calculations which are discussed in the next section. In this procedure a cartesian coordinate system is set up in a plane tangent to the spherical earth model, such that the origin is coincident with the position estimate produced by the iterative procedure. The lines of bearing are assumed to be straight lines in this plane.

A single representative line of bearing is shown in Figure 14 with (x_o, y_o) being the true target location.

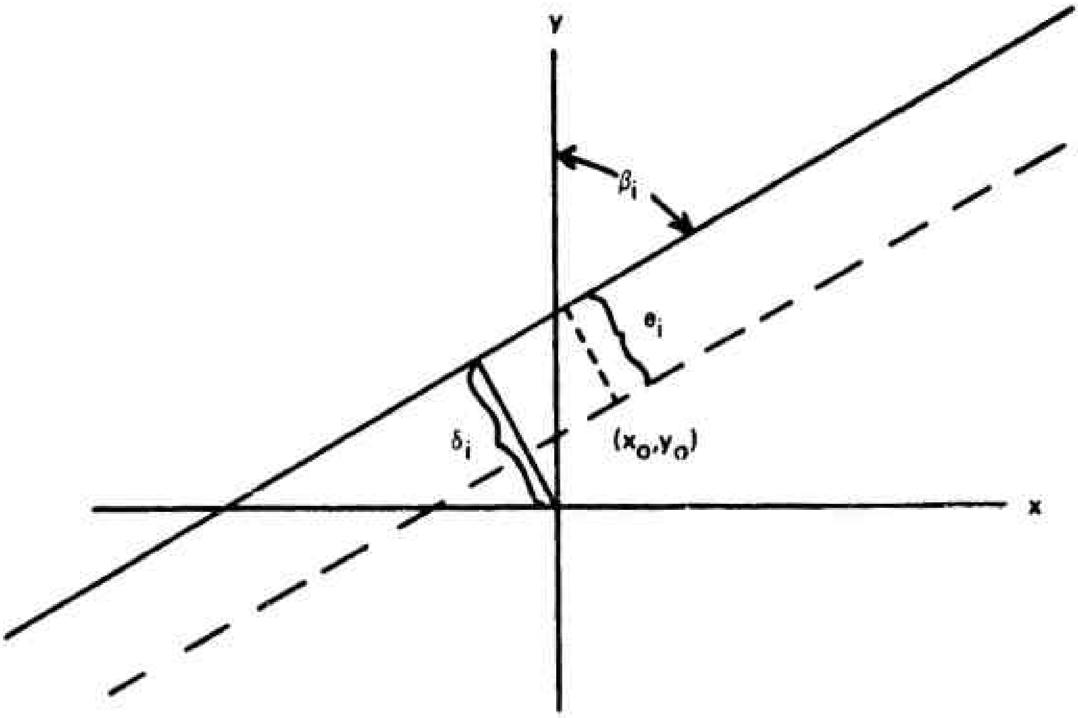


Figure 14. Coordinate System for the REFIX/2 Procedure

From elementry algebra, the equation of the solid line in Figure 14 is $(-\cos \beta_i)x + (\sin \beta_i)y = \delta_i$. The equation of the dashed line through

the true target location (x_o, y_o) is $(-\cos \beta_i)x + (\sin \beta_i)y = \delta_i - e_i$

Then, since (x_o, y_o) is on the dashed line, we have:

$$\delta_i = (-\cos \beta_i)x_o + (\sin \beta_i)y_o - e_i$$

This equation is in the form of a general multiple regression model. Both the β_i and the δ_i can be calculated since the origin is known. The β_i is approximately the bearing from the origin to the station for distant stations and approximately the reported bearing for near stations. The δ_i is approximately $\gamma_i \sin d_i$ where γ_i is the angular difference between the reported bearing and the bearing from the station to the origin, and where d_i is the distance from the station to the origin. Also, e_i, the perpendicular distance from the true target location to the bearing, is a normal random variable since $e_i = \varepsilon_i \sin d_i$ and ε_i (the angular error) is a normal random variable. Furthermore, since the ε_i are normal and independent with mean zero and variance σ_i^2, the e_i will be normal and independent with mean zero and variance $(\sin^2 d_i)\sigma_i^2$. Thus, the problem reduces to the simple multiple regression problem of estimating $\underline{b}$ when

$$\underline{\delta} = \underline{A}\underline{b} + \underline{e}$$

where

$$\underline{\delta} = \begin{bmatrix} \delta_1 \\ \cdot \\ \cdot \\ \cdot \\ \delta_k \end{bmatrix} \qquad A = \begin{bmatrix} -\cos \beta_1 & \sin \beta_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ -\cos \beta_k & \sin \beta_k \end{bmatrix}$$

$$\underline{b} = \begin{bmatrix} x_o \\ y_o \end{bmatrix} \text{ and } \underline{e} = \begin{bmatrix} e_1 \\ \cdot \\ \cdot \\ \cdot \\ e_k \end{bmatrix}$$

Further, we have that $\underline{e}$ is distributed as a multivariate random variable with $\underline{\mu} = \underline{0}_{k\times 1}$ and diagonal covariance matrix

$$\Sigma = \sigma_o^{\,2} \begin{bmatrix} \sin^2 d_1 w_1^{\,2} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \sin_2^2 w_2^2 & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & & \sin^2 d_k w_k^{\,2} \end{bmatrix} = \sigma_o^{\,2}\, W$$

with $\sigma_i^{\,2} = (w_i \sigma_o)^2$ for each station.

Using the above notation, the maximum likelihood estimate of $\underline{b} = (x_o, y_o)$, the true target location, is given by

$$\hat{\underline{b}} = (A^T \ \Sigma^{-1} A)^{-1} A^T \ \Sigma^{-1} \underline{\delta}$$

Once $\hat{\underline{b}}$ is available, we update the target vector to give the refined position estimate.

Relative Advantages. Either termination procedure works quite adequately in practice. The REFIX/1 procedure has a slight advantage when the position estimate is very close to one station since the assumption of parallel movement of bearings which is made in REFIX/2 is not really true in this case. On the other hand, REFIX/2 is quicker in some cases since on occasion REFIX/1 requires a number of additional iterations.

The Confidence Region

The calculation of the confidence region follows directly from the REFIX/2 algorithm. The estimate $\hat{\underline{b}}$ is distributed multivariate normal with mean $\underline{\mu}^* = [x_o, y_o]^T$ and covariance matrix

$$\Sigma^* = (A^T \ \Sigma^{-1} \ A)^{-1} = \frac{1}{\sigma_o^2} (A^T \ W^{-1} \ A)^{-1}$$

The minimum area 100 $(1-\alpha)$% confidence region is given by

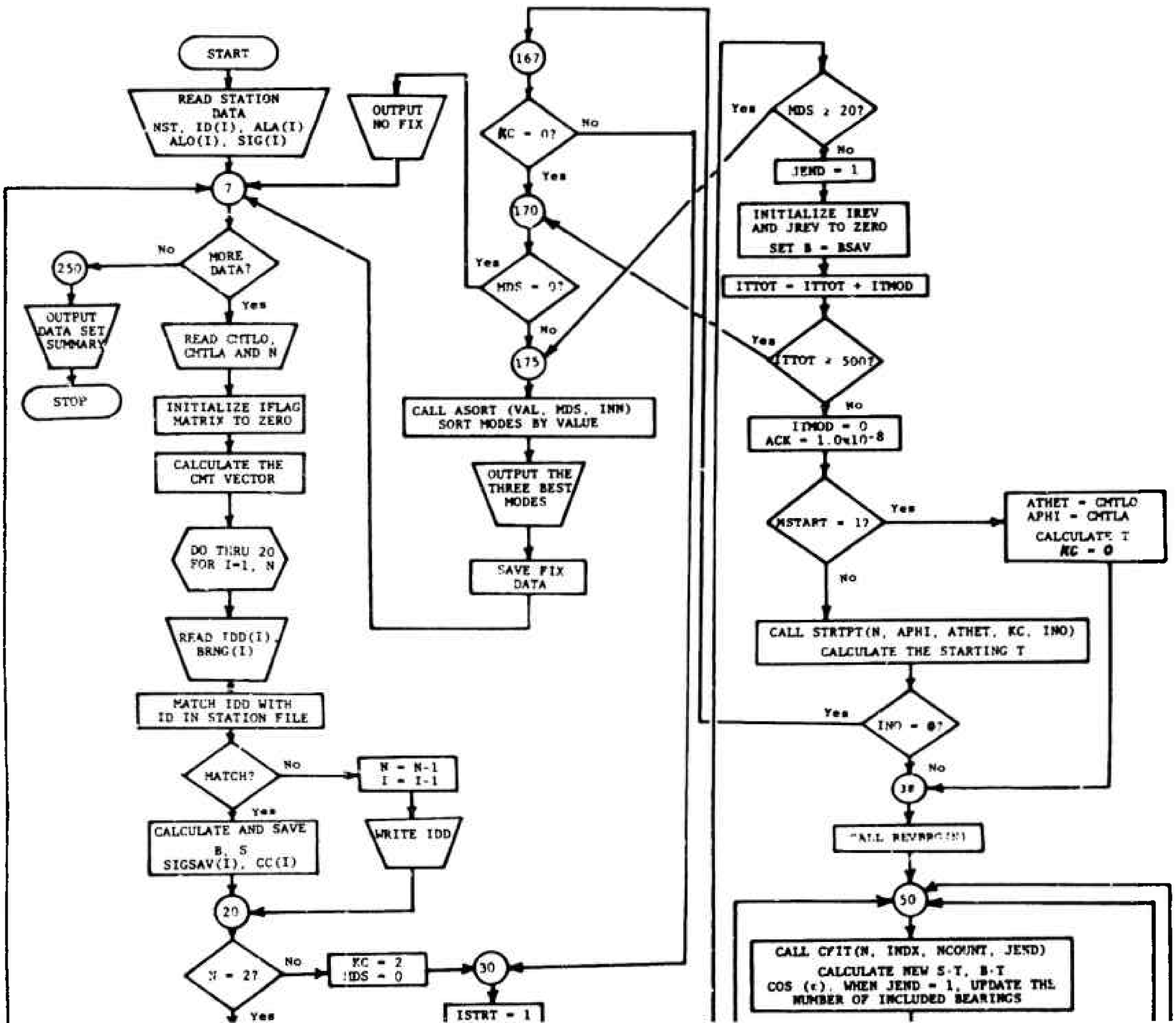
$$\frac{1}{s_o^2} [x, y] \ \Sigma^* \begin{bmatrix} x \\ y \end{bmatrix} \leq F_{\alpha,2,n-2}$$

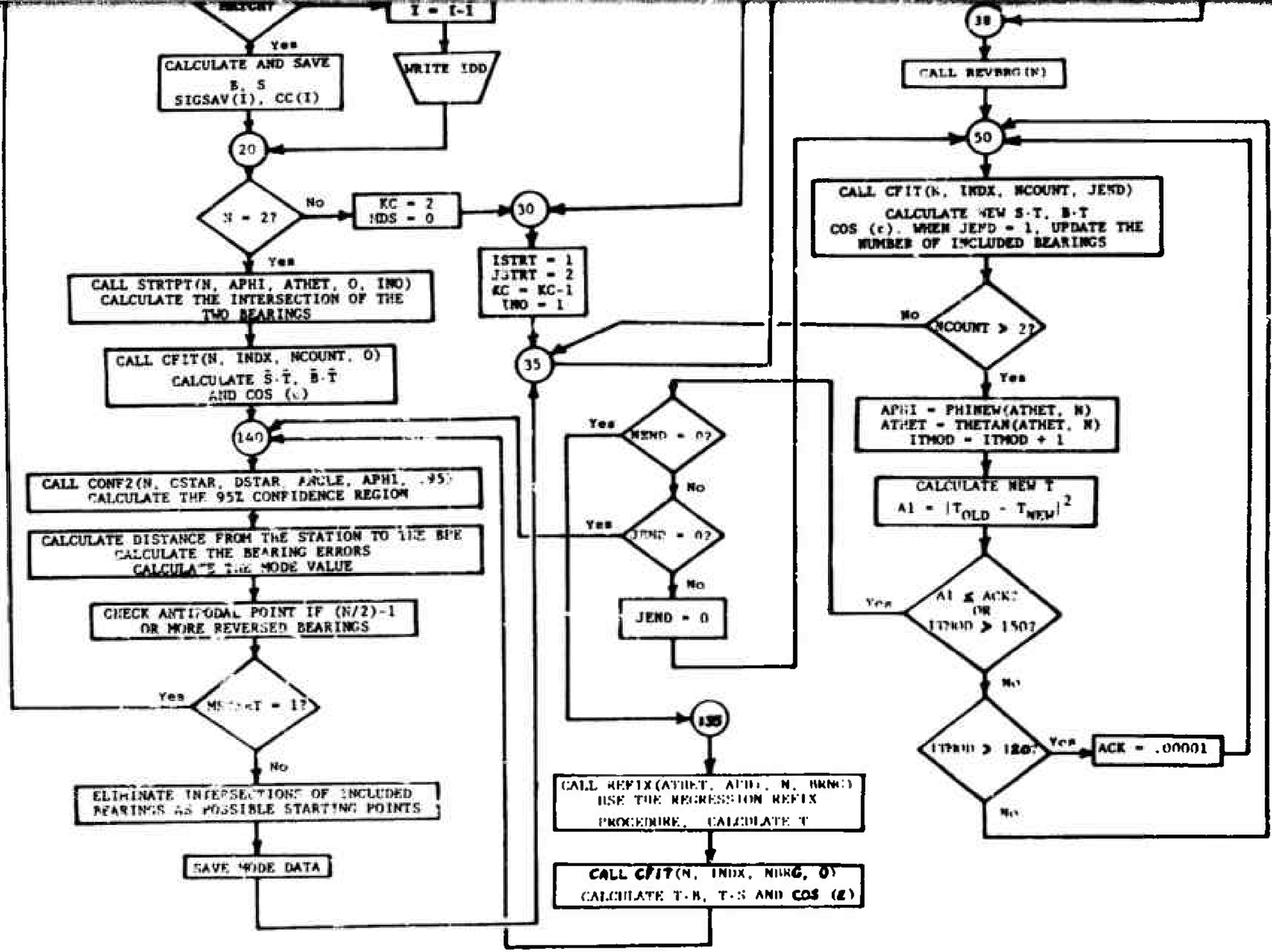
where s_o^2 is the estimate of σ_o^2 based on n data points and $F_{\alpha,2,n-2}$ in the appropriate F distribution percentage point. In this algorithm, the σ_o^2 is assumed known so $F_{\alpha,2,\infty}$ is used. This value is approximated by $-\log_e(\alpha)$.

The ellipse which is generated is given by the direction of the major axis and the magnitudes of the semi-major and semi-minor axes.

Appendix A

Computer Program Flow Chart


START
READ STATION DATA NST, ID(I), ALA(I) ALO(I), SIG(I)
7
MORE DATA?
No
Yes
250
OUTPUT DATA SET SUMMARY
STOP
READ CMTLO, CMTLA AND N
INITIALIZE IFLAG MATRIX TO ZERO
CALCULATE THE CMT VECTOR
DO THRU 20 FOR I=1, N
READ IDD(I), BRNG(I)
MATCH IDD WITH ID IN STATION FILE
MATCH?
N = N-1
I = I-1
WRITE IDD
CALCULATE AND SAVE B, S SIGSAV(I), CC(I)
20
N = 2?
KC = 2
MDS = 0
30
ISTRT = 1
OUTPUT NO FIX
167
KC = 0?
170
MDS = 0?
175
CALL ASORT (VAL, MDS, INN) SORT MODES BY VALUE
OUTPUT THE THREE BEST MODES
SAVE FIX DATA
MDS ≥ 20?
JEND = 1
INITIALIZE IREV AND JREV TO ZERO
SET B = BSAV
ITTOT = ITTOT + ITMOD
ITTOT ≥ 500?
ITMOD = 0
ACK = 1.0x10-8
NSTART = 1?
ATHET = CMTLO
APHI = CMTLA
CALCULATE T
KC = 0
CALL STRTPT(N, APHI, ATHET, KC, INO)
CALCULATE THE STARTING T
INO = 0?
38
50
CALL CFIT(N, INDX, NCOUNT, JEND)
CALCULATE NEW S·T, B·T
COS (ε), WHEN JEND = 1, UPDATE THE NUMBER OF INCLUDED BEARINGS



CALCULATE AND SAVE B, S SIGSAV(I), CC(I)
I = I-1
WRITE IDD
20
N = 2?
KC = 2 NDS = 0
30
ISTRT = 1 JSTRT = 2 KC = KC-1 INO = 1
35
CALL STRTPT(N, APHI, ATHET, 0, INO) CALCULATE THE INTERSECTION OF THE TWO BEARINGS
CALL CFIT(N, INDX, NCOUNT, 0) CALCULATE S̄·T̄, B̄·T̄ AND COS (c)
140
CALL CONF2(N, CSTAR, DSTAR, ANGLE, APHI, .95) CALCULATE THE 95% CONFIDENCE REGION
CALCULATE DISTANCE FROM THE STATION TO THE BPE CALCULATE THE BEARING ERRORS CALCULATE THE MODE VALUE
CHECK ANTIPODAL POINT IF (N/2)-1 OR MORE REVERSED BEARINGS
ELIMINATE INTERSECTIONS OF INCLUDED BEARINGS AS POSSIBLE STARTING POINTS
SAVE MODE DATA
38
CALL REVBRG(N)
50
CALL CFIT(N, INDX, NCOUNT, JEND) CALCULATE NEW S·T, B·T COS (c). WHEN JEND = 1, UPDATE THE NUMBER OF INCLUDED BEARINGS
NCOUNT > 2?
APHI = PHINEW(ATHET, N) ATHET = THETAN(ATHET, N) ITMOD = ITMOD + 1
CALCULATE NEW T A1 = |T_OLD - T_NEW|^2
A1 ≤ ACK? OR ITMOD > 150?
ITMOD > 120?
ACK = .00001
NEND = 0?
JEND = 0?
JEND = 0
135
CALL REFIX(ATHET, APHI, N, BRNG) USE THE REGRESSION REFIX PROCEDURE, CALCULATE T
CALL CFIT(N, INDX, NBRG, 0) CALCULATE T·B, T·S AND COS (c)
Yes
No

Appendix B

Computer Program Listing

```
$RESET FREE
FILE  3=HFDF/STATION,UNIT=DISK,RECORD=14,BLOCKING=30
C++ FILE 3 CONTAINS STATION INFORMATION.  FORMAT: CARD #1 HAS
C++        THE NUMBER OF STATIONS, NST, USING I5.  CARDS #2 THROUGH
C++        #NST+1 CONTAIN THE STATION IDENTIFIER, STATION LATITUDE,
C++        STATION LONGITUDE, AND STATION STANDARD DEVIATION USING
C++        (A2,3F10.0) AS THE INPUT FORMAT.  LATITUDE AND LONGITUDE
C++        ARE IN DECIMAL DEGREES WITH NEGATIVE BEING WEST LONGITUDE
C++        OR SOUTH LATITUDE.
FILE  7=OUTP,UNIT=PRINTER,RECORD=22
C++  FILE 7 IS AN OUTPUT FILE.
FILE  4=HFDF/IN2,UNIT=DISK,RECORD=14,BLOCKING=30
C++  FILE 4 IS THE CASE INPUT FILE.  DATA IS SET UP AS FOLLOWS:
C++          CARD #1 (12A6)- A 72 CHARACTER TITLE.
C++          CARD #2 (I2,2F6.1)- NUMBER OF BEARINGS REPORTING, AND DMI
C++               LATITUDE AND LONGITUDE, OR THE LATITUDE AND LONGITUDE OF
C++               THE OPTIONAL MANUAL STARTING POINT (LATITUDE AND LONGI-
C++               TUDE INPUT IN DECIMAL DEGREES).  A SECOND OPTION INPUTS
C++               LATITUDE AND LONGITUDE IN DEGREES AND MINUTES (E.G.,
C++               72.15 MEANS 72 DEGREES AND 15 MINUTES) USING AN
C++               (I2,F6.2,F8.2) FORMAT.
C++          CARDS #3+ (A2,F10.0)- STATION ID, REPORTED BEARING.
C++          REPEAT CARDS #1 THROUGH #3+ FOR EACH CASE.
C
C
C ************** F A L C O N F I X ***********************************
C
C++ CURRENT VERSION  79176
C
C++ FALCONFIX IS A FIX COMPUTATION ALGORITHM DEVELOPED BY
C++ LT COL WILLIAM T. HODSON AND MAJOR JOSEPH C. H. SMITH OF
C++ THE DEPARTMENT OF MATHEMATICAL SCIENCES, U. S. AIR FORCE
C++ ACADEMY, COLORADO 80840 (AVN 259-4470).  THIS ALGORITHM WAS
C++ DEVELOPED UNDER THE JOINT SPONSORSHIP OF THE NAVAL SECURITY
C++ GROUP COMMAND AND THE USAF ACADEMY.  IT IS INTENDED FOR USE IN
C++ THIS COUNTRY'S WORLD-WIDE SYSTEM OF HIGH FREQUENCY DIRECTION
C++ NETS.  ADDITIONAL DOCUMENTATION IS PROVIDED IN USAF ACADEMY
C++ TECHNICAL REPORT TR 79-5.  SPECIFIC PAGE REFERENCES IN THE TECH
C++ REPORT ARE GIVEN IN BRACKETS THROUGHOUT THE COMMENTS BELOW (E.G.,
C++ [TR 79-5:13] REFERS TO PAGE 13 IN THE TECH REPORT.)
C++ THE FOLLOWING COMMENT CARD WILL APPEAR THROUGHOUT THE PROGRAM:
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARD$$
C++ THIS CARD MARKS POINTS IN THE PROGRAM IN WHICH VARIABLES HAVE BEEN
```

```
C++ ASSIGNED SPECIFIC VALUES.  THE USER MAY CHANGE THESE VALUES IF
C++ OPERATING CONDITIONS INDICATE THAT THE CURRENT VALUES ARE NOT
C++ APPROPRIATE.
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C++ VARIABLE DEFINITIONS:
C++ COMMON/STAT/: STATION PARAMETERS WITH NST THE NUMBER OF STATIONS,
C      ID THE STATION TWO LETTER IDENTIFIER,ALA AND ALO STATION
C      LATITUDE AND LONGITUDE, AND SIG THE LONG RANGE STANDARD
C      DEVIATION OF ANGULAR ERROR IN DEGREES.
C++ COMMON/FLAG/: AREA REQUIRED TO KEEP TRACK OF BEARING INTERSEC-
C      TIONS AS A BASIS OF DETERMINING STARTING POINTS FOR THE ITERATIVE
C      PROCEDURE.
C++ COMMON/SAVE/.  THIS AREA IS USED TO KEEP TRACK OF THE DATA FOR
C      EACH INDIVIDUAL BEST POINT ESTIMATE(BPE) THE AREA IS USED FOR
C      OUTPUT ONCE ALL POSSIBLE BPE'S HAVE BEEN FOUND.  LAD,LAM,LAS,
C      AND LOD,LOM,LOS ARE DEGREES-MINUTES-SECONDS FOR BPE LATITUDE AND
C      AND LONGITUDE.  MNS AND MEW SPECIFY NORTH-SOUTH AND EAST-WEST FOR
C      EACH BEST POINT ESTIMATE(INTERNAL TO THE PROGRAM THE CONVENTION
C      THAT SOUTH AND WEST ARE NEGATIVE IS USED).  AA2,AXES2,AND ADEG
C      ARE THE SEARCH AREA, MAJOR AND MINOR AXES, AND MAJOR AXIS
C      ORIENTATION FOR THE CONFIDENCE ELLIPSE BASED ON THE ASSUMPTION
C      THAT THE VARIANCES ARE KNOWN.  VAL IS THE VALUE ASSOCIATED WITH
C      EACH BPE.(TR 79-5:22).  INS SPECIFIES THE INCLUDED BEARINGS,
C      DISI IS THE DISTANCE IN NAUTICAL MILES FROM THE STATION
C      TO THE BPE, ERR IS THE ABSOLUTE VALUE OF THE ANGULAR ERROR FOR THE
C      REPORTED BEARING.  IDD AND BRNG ARE THE STATIONS REPORTING AND THE
C      REPORTED BEARINGS.  IABT IS NOT USED IN THIS VERSION.
C++ COMMON/ITR/: AREA USED TO KEEP TRACK OF VALUE DURING EACH ITERATION.
C      B - BEARING VECTORS; S - STATION VECTORS; T - CURRENT BPE VECTOR;
C      DST - S DOT T; DBT - B DOT T; FIT - COSINE(ERROR)/VARIANCE;
C      D - SQUARED SINE OF THE ANGLE BETWEEN THE BPE AND THE STATION;
C      ASIG - CURRENT ANGULAR STANDARD DEVIATION; INN - BEARINGS INCLUDED
C      IN THE CURRENT CALCULATIONS.  BSAV AND SIGSAV ARE THE ORIGINAL
C      B-VECTORS AND THE STANDARD DEVIATION AT 1000 KM. IREV IS USED TO
C      KEEP TRACK OF REVERSED BEARINGS.
C++ COMMON/SUMARY/:  SUMMARY DATA.  THE DIMENSION OF 300 LIMITS THE
C      NUMBER OF BEARING SETS PER RUN TO 300.  ITSUM SAVES THE FIRST 12
C      CHARACTERS OF THE CASE TITLE.  NMSUM IS THE NUMBER OF MODES FOUND.
C      BSUM IS BEARING TO THE CMT.  CSUM IS THE DISTANCE TO THE CMT.
C      DSUM IS THE DISTANCE TO THE CONFIDENCE ELLIPSE IN THE DIRECTION OF
C      THE CMT.  ESUM IS THE SEARCH AREA.  FSUM IS THE VALUE USED FOR
```

```
007500 C     ORDERING THE MODES.  HSUM IS THE SIZE OF THE COCKED HAT.
007700 C     LSUM IS THE NUMBER OF BEARINGS USED IN THE FIX.
007800 C     THE ABOVE ARE SAVED FOR THE BEST THREE MODES OF EACH CASE.
007900 C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
008000 C++++++++++ INDEX TO SUBROUTINES AND FUNCTIONS +++++++++++++++++++++++++
008100 C
008200 C   SUBROUTINES
008300 C     BEGINNING SEQ #          NAME
008400 C          57900          ASORT(X,N,IORD)
008500 C          62200          CFIT(N,INC,NBRG,NINC)
008600 C          65900          CONF2(N,CSTAR,DSTAR,ANGLE,PHI,P0)
008700 C          70200          CROSS(A,B,C)
008800 C          71200          DMS(AA,AD,AM,AS,IH,IUPT)
008900 C          74100          FLAGCK(N,IN,JN)
009000 C          76400          FLAGS(INN,INC)
009050 C          82100          REFIX(THETA,PHI,NBRG,BRNG)
009100 C          88200          REVBRG(N)
009200 C          93300          STRTPT(N,PHI,THETA,KC,IND)
009300 C
009400 C   FUNCTIONS
009500 C     BEGINNING SEQ #          NAME
009600 C          56700          ARCCOS(A)
009700 C          57400          ARCSIN(A)
009750 C          60100          AVAL(N,MDS,DIST,ERR,INN,JREV)
009800 C          73500          DDT(A,L)
009900 C          77300          FSIG(SU,SIGM,SDT)
010000 C          80200          PHINEW(A,N)
010100 C          90300          SIGC(SU,SIGM,SDT)
010200 C          100000         THETAN(A,N)
010300 C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
010400 C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
010500       COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FIT(20),D(20)
010600      *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
010700       COMMON/STAT/JST,ID(27),ALA(27),ALO(27),CCC(27),SIG(27)
010800       COMMON/FLAG/IFLAG(20,20),ISTRT,JSTRT
010900       COMMON/SAVE/LAD(20),LAM(20),LAS(20),LOD(20),LDM(20),LDS(20),
011000      +MNS(20),MEW(20),AA2(20),AXES2(20,2),VAL(20),
011100      +NABT(20),INS(20,20),DIST(20,20),ERR(20,20),IDD(20),BRNG(20),ADEG(2
011200      +0),ITITLE(12)
011300       COMMON/SUMMARY/ITSUM(300,2),NMSUM(300),              BSUM(300,3),CSU
011400      *M(300,3),DSUM(300,3),ESUM(300,3),LSUM(300,3),              FSUM(300,
011500      *3),HSUM(300,3)
011600       DIMENSION INC(3),TT(3),TS(3),INDX(20),IHEADR(4)
```

```
      DATA INDX/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20/
      DATA IHEADR/'INC','DISIAN','ICE','ERROR'/
C++R2D AND D2R CONVERT FROM RADIANS TO DEGREES AND VICE VERSA.
      R2D=57.29577951
      D2R=.01745329З
C+++++++ OPTION CONTROLS++++++++++++++++++++++++++++
C+++(REPLACE HERE AND IN SUBROUTINE STRTPT)+++++++++
C++  LW IS THE OUTPUT FILE: 7-LINE PRINTER,6-REMOTE TERMINAL
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      LW=7
C++  KW=0 LIMITS OUTPUT TO ESSENTIAL INFORMATION.
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      KW=0
C++  MSTART IS A MANUAL START POINT OPTION.  IF MSTART IS
C++       SET TO 1, CMTLAT AND CMTLONG INPUT ON CARD #2 WILL
C++       BE USED AS THE STARTING POINT, AND ONLY ONE MODE
C++       WILL BE CALCULATED.
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      MSTART=0
C++  KWCMT=1 LIMITS OUTPUT WHEN CMT INFORMATION IS NOT AVAILABLE.
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      KWCMT=1
C++NEND PROVIDES A TERMINATION OPTION.  IF NEND IS 0, THE PROGRAM
C++TERMINATES BY USING THE REGRESSION FIX COMPUTATION.  IF NEND IS
C++ 1, THE PROGRAM TERMINATES BY ITERATING USING A CONSTANT BEARING
C++SET AND VARIANCE.
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      NEND=0
C++
C+++++++ END OF OPTION CONTROLS ++++++++++++++++++++++++++++++++++++++
C++READ IN STATION INFORMATION
      READ(3,7004)NST
      DO 5 I=1,NST
    5 READ(3,7006)ID(I),ALA(I),ALO(I),SIG(I)
C++SIG IS INPUT IN DEGREES.
 7004 FORMAT(I5)
 7006 FORMAT(A2,3E10.8,I5)
C++READ CASE TITLE AND BEARING INFORMATION FOR INDIVIDUAL CASE.
C++NCASE COUNTS THE NUMBER OF CASES, ITTOT COUNTS THE TOTAL NUMBER OF
C++ITERATIONS FOR EACH CASE.
      NCASE=0
    7 ITTOT=0
      IF(KW.EQ.1)WRITE(LW,7075)
```

```
 7075 FORMAT(1H1)
      READ(4,7010,END=250)ITITLE
 7010 FORMAT(12A6)
C++NRODEC=0 IS THE OPTION FOR CMTLA AND CMTLO TO BE INPUT IN DEGREES AND
C++MINUTES(72.15 MEANS 72 DEGREES AND 15 MINUTES).  NRODEC=1 IS THE
C++OPTION FOR CMTLA, CMTLO TO BE INPUT AND DEGREES AND DECIMAL DEGREES.
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARD$$
      NRODEC=0
      IF(NRODEC.EQ.0)GO TO 8
      READ(4,6111)N,CMTLA,CMTLO
 6111 FORMAT(I2,2F6.1)
      GO TO 9
    8 READ(4,7001)N,CMTLA,CMTLO
 7001 FORMAT(I2,F6.2,F8.2)
C++BEGINNING OF PROCEDURE TO CONVERT DEGREES AND MINUTES TO DECIMAL
C++DEGREES.
      STOR=ABS(CMTLO)
      INT=STOR
      FRAC=(STOR-INT)*100.
      CMTLO=SIGN(INT+FRAC/60.,CMTLO)
      STOR=ABS(CMTLA)
      INT=STOR
      FRAC=(STOR-INT)*100.
      CMTLA=SIGN(INT+FRAC/60.,CMTLA)
C++END OF PROCEDURE TO CONVERT DEGREES AND MINUTES TO DECIMAL DEGREES.
C++INITILIZE THE STARTING POINT ARRAY.
    9 DO 13 II=1,N
      DO 13 JJ=1,N
   13 IFLAG(II,JJ)=0
C++CALCULATE THE CMT VECTOR.
      A=D2R*CMTLA
      BB=D2R*CMTLO
      CMT1=COS(BB)*COS(A)
      CMT2=SIN(BB)*COS(A)
      CMT3=SIN(A)
      DO 22 I=1,N
      READ(4,7007)IDD(I),BRNG(I)
 7007 FORMAT(A2,F10.9)
C++MATCH REPORTED BEARING WITH STATION REPORTING.  CALCULATE THE
C++STATION-VECTOR(S) AND THE BEARING-VECTOR(B)
      DO 15 J=1,NST
      IF(IDD(I).NE.ID(J))GO TO 15
      A=ALA(J)
```

```
      BB=ALO(J)
      SIGSAV(I)=SIG(J)
      CC(I)=CCC(I)
      GO TO 20
   15 CONTINUE
C++   IF YOU GET TO HERE THE STATION REPORTING THE BEARING IS
C++   NOT IN THE STATION FILE.  THE PROGRAM DELETES THE BEARING
C++   AND PROCEEDS.
      WRITE(LW,7000)IDD(I)
 7000 FORMAT('  STATION ',A2,' IS NOT IN THE STATION FILE.'//)
      I=I-1
      N=N-1
      GO TO 22
   20 A=D2R*A
      BB=D2R*BB
      C=D2R*BRNG(I)
C++CALCULATE THE STATION VECTOR AND THE BEARING VECTOR FOR THE
C++STATION
      SIGSAV(I)=D2R*SIGSAV(I)
      S(1,I)=COS(BB)*COS(A)
      S(2,I)=SIN(BB)*COS(A)
      S(3,I)=SIN(A)
      BSAV(1,I)=-(SIN(A)*COS(BB)*COS(C)+SIN(BB)*SIN(C))
      BSAV(2,I)=COS(BB)*SIN(C)-SIN(A)*SIN(BB)*COS(C)
      BSAV(3,I)=COS(A)*COS(C)
   22 CONTINUE
      IF(N.NE.2)GO TO 27
C++IF YOU GET TO HERE, YOU ONLY HAVE TWO STATIONS REPORTING.
C++CALCULATE THE INTERSECTION AND REPORT IT AS A BPE.
      ISTRT=1
      JSTRT=2
      DO 25 I=1,N
      IREV(I)=0
      ASIG(I)=SIGSAV(I)
      DO 25 J=1,3
   25 B(J,I)=BSAV(J,I)
      CALL STRTPT(N,APHI,ATHET,0,INO)
      CALL CFIT(N,INDX,NCOUNT,0)
      MDS=1
      INN(1)=1
      INN(2)=1
      GO TO 140
C++ KC IS A PARAMETER USED TO PREVENT STARTING AT BEARING
```

```
023300 C++ INTERSECTIONS WITH LESS THAN A 20 DEGREE INTERSECTION
023400 C++ ANGLE THE FIRST TIME THROUGH.  IF KC=3 IT WILL START WITH
023500 C++ 40 DEGREES, THEN USE 20 DEGREES, THEN PICK UP THE REMAIN-
023600 C++ DER.  IN GENERAL, IF KC=N, IT WILL START WITH (N-1)*20 DEGREES,
023602 C++ THEN USE (N-2)*20 DEGREES, AND PROCEED TU ZERO DEGREES.
023610 C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
023700    27 KC=2
023800 C++ MDS=NUMBER OF MODES
023900       MDS=0
024000 C++ RETURN TO LABEL 30 TO PICK UP NEW STARTING POINTS WITH KC
024100 C++ REDUCED
024200    30 ISTRT=1
024300       JSTRT=2
024400       KC=KC-1
024500       IND=1
024600 C++IND IS SET TO 0 WHEN ALL STARTING POINTS HAVE BEEN USED UP.
024700 C++ A NEW MODE CALCULATION BEGINS AT LABEL 35
024710 C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
024720 C++THE FOLLOWING STATEMENT LIMITS THE NUMBER OF MODES TO 20.
024800    35 IF(MDS.GE.20)GO TO 175
024810 C++ JEND IS A CONTROL TO ALLOW USE OF A CONSTANT BEARING SET AND
024820 C++CONSTANT VARIANCES IN THE ITERATION PROCEDURE WHEN SET TO ZERO.
024830       JEND=1
024835       IAPOD=0
024840 C++IAPOD AND NEWMD ARE INDICATORS TO ROUTE THE PROGRAM THROUGH THE
024845 C++ANTIPODAL POINT PROCEDURE LNS 35900-40500.
024850       NEWMD=0
025000 C++INITIALIZE THE ARRAY TO INDICATE IF BEARINGS HAVE BEEN REVERSED.
025100       DO 36 I=1,N
025200       IREV(I)=0
025210       JREV(I)=0
025300 C++ INITIALIZE BEARING VECTORS (SOME MAY HAVE BEEN REVERSED).
025400       DO 36 J=1,3
025500    36 B(J,I)=BSAV(J,I)
025600 C++ ITTOT COUNTS THE TOTAL NUMBERS OF ITERATIONS FOR THIS BEARING SET.
025700 C++ IF ITTOT IS GREATER THAN 500 STOP.
025800       ITTOT=ITTOT+ITMOD
025810 C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
025820 C++THE FOLLOWING STATEMENT LIMITS THE TOTAL NUMBER OF ITERATIONS TO 500.
025900       IF(ITTOT.GT.500)GO TO 170
026000 C++ ITMOD COUNTS THE NUMBER OF ITERATIONS FOR THIS MODE.  ACK IS THE
026100 C++ LIMIT USED TO TERMINATE THE ITERATIONS.  IF ITMOD IS GREATER THAN
026200 C++ 120, ACK IS REDUCED.  IF ITMOD IS GREATER THAN 150, THE ITERATION
```

```
C++ IS TERMINATED.
      ITMOD=0
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$$
C++THE FOLLOWING STATEMENT INITIALIZES THE STOPPING CRITERION FOR THE
C++ITERATIVE PROCEDURE.
      ACK=.00000001
C++ FIND A STARTING POINT
      IF(MSTART.NE.1)GO TO 37
C++ IF MSTART=1 USE CMTLA AND CMTLO AS THE STARTING POINT.
      ATHET=D2R*CMTLO
      APHI=D2R*CMTLA
      T(1)=COS(ATHET)*COS(APHI)
      T(2)=SIN(ATHET)*COS(APHI)
      T(3)=SIN(APHI)
      KC=0
      GO TO 38
   37 CALL STRTPT(N,APHI,ATHET,KC,IND)
C++THE SUBROUTINE STRTPT RETURNS THE NEXT STARTING POINT.
C++ IF IND=0 ALL STARTING POINTS HAVE BEEN USED
      IF(IND.EQ.0)GO TO 167
      APHID=R2D*APHI
      ATHETD=R2D*ATHET
   38 IF(KW.EQ.1)WRITE(LW,7072)APHID,ATHETD
 7072 FORMAT('  STARTING LALO.',2F10.3)
C++ REVERSE BEARINGS IF NECESSARY
      CALL REVBRG(N)
C
C++A NEW ITERATION BEGINS AT LABEL 50.
C++ CFIT UPDATES THE ARRAYS IN COMMON/ITR/
   50 CALL CFIT(N,INDX,NCOUNT,JEND)
C++ NCOUNT IS THE NUMBER OF USABLE BEARINGS.  IF <3 DO
C++ NOT ITERATE
      IF(NCOUNT.GT.2)GO TO 60
      IF(KW.EQ.1)WRITE(LW,7071)
 7071 FORMAT('  ITERATION STOPPED BECAUSE FEWER THAN TWO ACTIVE BEARINGS
     +.')
      GO TO 35
C++ CALCULATE THE NEW LATITUDE (APHI) AND LONGITUDE (ATHET)
   60 APHI=PHINEW(ATHET,N)
      ITMOD=ITMOD+1
      ATHET=THETAN(ATHET,N)
      APHID=R2D*APHI
      ATHETD=R2D*ATHET
```

```
      IF(KW.EQ.1)WRITE(LW,7073)ITMOD,APHID,ATHETD,(INN(I),I=1,N)
 7073 FORMAT(5X,'  ITERATION # ',I3,'  LALU:',2F10.3,5X,25I2)
C++ CALCULATE THE NEW T VECTOR, MAGNITUDE OF THE CHANGE, AND
C++ TEST FOR STOPPING THE ITERATIONS
      T1=COS(ATHET)*COS(APHI)
      T2=SIN(ATHET)*COS(APHI)
      T3=SIN(APHI)
      A1=T1-T(1)
      A2=T2-T(2)
      A3=T3-T(3)
C++VARIABLE AI BELOW IS THE MAGNITUDE OF THE DIFFERENCE BETWEEN THE NEW
C++ T-VECTOR AND THE OLD T-VECTOR.
      AI=A1*A1+A2*A2+A3*A3
C++REPLACE THE OLD T-VECTOR WITH THE NEW T-VECTOR.
      T(1)=T1
      T(2)=T2
      T(3)=T3
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARD$$
C++THE STATEMENT BELOW LIMITS THE MAXIMUM NUMBER OF ITERATIONS FOR ANY
C++ONE MODE TO 150.
      IF(AI.LE.ACK.OR.ITMOD.GT.150)GO TO 130
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARD$$
C++THE FOLLOWING STATEMENT REDUCES THE STOPPING CRITERION FOR THE
C++ITERATIVE PROCEDURE WHEN THE NUMBER OF ITERATIONS EXCEEDS 120.
      IF(ITMOD.GT.120)ACK=.00001
      GO TO 30
C++THE ITERATIVE PROCEDURE HAS TERMINATED WHEN YOU GET TO LABEL 130.
C++INITIATE THE PROPER REFIX PROCEDURE DEPENDING UPON THE VALUE OF NEND.
C++ (TR 79-5:26)
  130 NABT(MOS+1)=IT400
C++ A BPE HAS BEEN FOUND.  CALCULATE VALUES FOR THE BPE AND SAVE
  131 IF(NEND.EQ.0)GO TO 135
C++IF JEND=0, THE ITERATIVE TERMINAL PROCEDURE IS COMPLETE.
      IF(JEND.EQ.0)GO TO 140
      IF(KW.EQ.1)WRITE(LW,5000)
 5000 FORMAT(' TERMINATION INITIATED ')
      JEND=0
      GO TO 30
  135 CALL REFIX(ATHET,APHI,N,BRNG)
      CALL CFIT(N,INDX,NBRG,0)
C++ MOS=MODE NUMBER
C++ A BEST POINT ESTIMATE HAS BEEN FOUND. CALCULATE AND SAVE THE
C++ NECESSARY INFORMATION.
```

```
  140 CALL CONF2(N,CSTAR,DSTAR,ANGLE,APHI,.95)
      MDS=MDS+1
C++ CALCULATE THE DISTANCE FROM THE STATION TO THE BPE
      DO 150 I=1,N
C++INS SAVES THE INCLUDED BEARINGS ARRAY.
      INS(MDS,I)=INN(I)
      A=ARCSIN(SQRT(D(I)))
      IF(DST(I).LT.0)A=3.141592654-A
      DIST(MDS,I)=A*3444.
C++ IF THE DISTANCE IS GREATER THAN 90 DEGREES AND THE BEARING
C++ HAS BEEN REVERSED, CONSIDER IT AN AROUND THE WORLD BEARING
      IF(DIST(MDS,I).GT.5400.AND.IREV(I).EQ.1)DIST(MDS,I)=
     *21601.59-DIST(MDS,I)
C++  CALCULATE THE BEARING ERROR FOR EACH STATION
      DEMON=(T(3)-S(3,I)*DST(I))/SQRT((1-S(3,I)**2)*D(I))
      DEMON=ARCCOS(DEMON)*R2D
      IF((S(1,I)*T(2)-T(1)*S(2,I)).LT.0)DEMON=360-DEMON
      BRING=BRNG(I)
      IF(IREV(I).EQ.1)BRING=AMOD(BRING+180.,360.)
      ERR(MDS,I)=BRING-DEMON
      A=ABS(ERR(MDS,I))
      IF(A.GT.180)ERR(MDS,I)=-SIGN(360.-A,ERR(MDS,I))
  150 CONTINUE
C++SAVE INFORMATION FOR THIS MODE.
      CALL DMS(APHI,LAD(MDS),LAM(MDS),LAS(MDS),MNS(MDS),0)
      CALL DMS(ATHET,LOD(MDS),LOM(MDS),LOS(MDS),MEN(MDS),1)
      AA2(MDS)=CSTAR*DSTAR*3.1416
      AXES2(MDS,1)=AMAX1(CSTAR,DSTAR)
      AXES2(MDS,2)=AMIN1(CSTAR,DSTAR)
      VAL(MDS)=AVAL(N,MDS,DIST,ERR,INN,JREV)
      ADEG(MDS)=R2D*ANGLE
      IF(KN.EQ.1)WRITE(LW,7074)LAD(MDS),LAM(MDS),LAS(MDS),MNS(MDS),LUD(M
     *DS),LOM(MDS),LOS(MDS),MEN(MDS),VAL(MDS)
 7074 FORMAT(' BPE:',2X,I2,'-',I2,'-',I2,A1,5X,I3,'-',I2,'-',I2,A1,
     110X,'  LIKELIHOOD=',F15.5//)
C++PROCEDURE TO INCLUDE THE ANTIPODAL POINT IF THERE ARE
C++(N/2)-1 OR MORE REVERSED BEARINGS(LNS 35900-40500).
C++ I1 COUNTS THE INCLUDED BEARINGS,I2 COUNTS THE BEARINGS WHICH ARE
C++ INCLUDED AND REVERSED.  DISTSU IS THE SUM OF THE DISTANCES TO BPE.
      I1=0
      I2=0
      DISTSU=0.
      DO 153 I=1,N
```

```
      IF(INN(I).EQ.0)GO TO 153
      I1=I1+1
      DISTSU=DISTSU+DIST(MDS,I)
      IF(JREV(I).EQ.0)GO TO 153
      I2=I2+1
  153 CONTINUE
      IF(IAPOD.EQ.1)GO TO 156
      IAPOD=1
      JEND=1
      I3=I1/2.
      IF(I2.LT.I3)GO TO 165
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$$
C++IF THE MEAN DISTANCE FROM THE STATION TO THE BPE IS LESS THAN ONE-
C++FORTH OF THE WAY AROUND THE GLOBE(5400 NM) THE FOLLOWING STATEMENT
C++CAUSES THAT TO BE THE ONLY REPORTED BPE.
      IF(DISTSU/I1.LE.5400.)GO TO 165
      NEWMD=1
      APHI=-APHI
      ATHET=ATHET+3.141593
      IF(ATHET.GT.3.141593)ATHET=ATHET-6.283186
      T(1)=COS(ATHET)*COS(APHI)
      T(2)=SIN(ATHET)*COS(APHI)
      T(3)=SIN(APHI)
      DO 154 I=1,N
      IREV(I)=0
      JREV(I)=0
      DO 154 J=1,3
  154 B(J,I)=BSAV(J,I)
      CALL REVBRG(N)
      CALL CFIT(N,INDX,NBRG,0)
      GO TO 131
  156 IF(NEWMD.EQ.0)GO TO 165
      IF(DISTSU/I1.GT.5400.)GO TO 165
      MDS=MDS-1
      DO 158 I=1,N
      INS(MDS,I)=INS(MDS+1,I)
      DIST(MDS,I)=DIST(MDS+1,I)
  158 ERR(MDS,I)=ERR(MDS+1,I)
      LAD(MDS)=LAD(MDS+1)
      LAM(MDS)=LAM(MDS+1)
      LAS(MDS)=LAS(MDS+1)
      LOD(MDS)=LOD(MDS+1)
      LOM(MDS)=LOM(MDS+1)
```

```
      LOS(MDS)=LOS(MDS+1)
      AA2(MDS)=AA2(MDS+1)
      AXES2(MDS,1)=AXES2(MDS+1,1)
      AXES2(MDS,2)=AXES2(MDS+1,2)
      VAL(MDS)=VAL(MDS+1)
      ADEG(MDS)=ADEG(MDS+1)
      MNS(MDS)=MNS(MDS+1)
      MEW(MDS)=MEW(MDS+1)
  165 IF(MSTART.EQ.1)GO TO 170
C++ ELIMINATE INTERSECTIONS OF USED BEARINGS AS POSSIBLE STARTING
C++ POINTS
      CALL FLAGST(N,INN)
      GO TO 35
C++ INCREMENT KC.  IF KC=0 ALL MODES HAVE BEEN FOUND. STOP.
  167 IF(KC.NE.0)GO TO 30
C++ FIND THE THREE BEST BPE'S BASED ON THE NUMBER OF FORWARD
C++ BEARINGS AND THE HAT SIZE.
  170 IF(MDS.NE.0)GO TO 175
      WRITE(LW,7021)ITITLE
      CALL DMS(CMTLA*D2R,IAD,IAM,IAS,JNS,0)
      CALL DMS(CMTLO*D2R,IOD,IOM,IOS,IEW,1)
      IF(MSTART.NE.1)WRITE(LW,7035)N,IAD,IAM,IAS,JNS,IOD,IOM,IOS,IEW,MDS
 7035 FORMAT(/'    NUMBER OF BEARINGS: ',1X,I3,'    CMT LALO: ',I2,'-',I2,
     *'-',I2,A1,5X,I3,'-',I2,'-',I2,A1,5X,'NUMBER OF MODES: ',I2)
      IF(MSTART.EQ.1)WRITE(LW,7036)N,IAD,IAM,IAS,JNS,IOD,IOM,IOS,IEW
 7036 FORMAT(/'    NUMBER OF BEARINGS: ',1X,I3,'    MANUAL STARTING LALU: '
     *,I2,'-',I2,'-',I2,A1,5X,I3,'-',I2,'-',I2,A1)
      WRITE(LW,7009)
 7009 FORMAT(/// 20X,'N O    F I X ! ! ! !')
      WRITE(LW,7057)
 7057 FORMAT(//3X,'BEARINGS')
      DO 172 I=1,N
  172 WRITE(LW,7058)IDD(I),BRNG(I)
 7058 FORMAT(2X,A2,F8.1)
      GO TO 7
  175 CALL ASORT(VAL,MDS,INN)
      NPRT=MIN0(3,MDS)
      IF(NPRT.GT.1)GO TO 177
      INC(1)=100
      GO TO 215
C++   BEGINNING OF A ROUTINE TO ELIMINATE MODES WITH THE SAME BEARING
C++   SETS.  MULTIPLE MODES WITH THE SAME BEARING SET CAN OCCUR WHEN A
C++   STARTING POINT BEARING IS NOT IN THE FINAL BEARING SET.
```

```
  177 ISTOP=NPRT-1
      DO 205 I=1,ISTOP
      IP1=I+1
      DO 205 J=IP1,NPRT
      II=INN(I)
      JJ=INN(J)
      IS=0
      JS=0
      KS=0
      DO 185 K=1,N
      IS=IS+INS(II,K)
      KS=KS+INS(JJ,K)
      JS=JS+INS(II,K)*INS(JJ,K)
  185 CONTINUE
      IF(IS.NE.JS)GO TO 205
      IF(KS.NE.JS)GO TO 205
      IF(MEW(II).NE.MEW(JJ).AND.LOD(II)+LOD(JJ).LT.330.AND.
     *LOD(II)+LOD(JJ).GT.30)GO TO 205
      IF(MEW(II).EQ.MEW(JJ).AND.ABS(LOD(II)-LOD(JJ)).GT.30)GO TO 205
      MDS=MDS-1
      IF(MDS.NE.1)GO TO 190
      NPRT=1
      GO TO 215
  190 IF(MDS.LT.NPRT)NPRT=MDS
      IF(MDS.LT.J)GO TO 215
      DO 202 K=J,MDS
      INN(K)=INN(K+1)
  202 CONTINUE
      GO TO 177
  205 CONTINUE
C++++++++++++++++END OF ROUTINE TO ELIMINATE DUPLILATE MODES++++++++++++
  215 WRITE(LW,7021)ITITLE
 7021 FORMAT(1H1,///,1X,12A6)
      NCASE=NCASE+1
      DO 217 I=1,2
  217 ITSUM(NCASE,I)=ITITLE(I)
      CALL DMS(CMTLA*D2R,IAD,IAM,IAS,JNS,0)
      CALL DMS(CMTLO*D2R,IOD,IOM,IOS,IEW,1)
      IF(MSTART.NE.1)WRITE(LW,7035)N,IAD,IAM,IAS,JNS,IOD,IOM,IOS,IEW,MDS
      IF(MSTART.EQ.1)WRITE(LW,7036)N,IAD,IAM,IAS,JNS,IOD,IOM,IOS,IEW
      NMSUM(NCASE)=NPRT
      DO 218 I=1,NPRT
      II=INN(I)
```

```
      A=LAD(II)+LAM(II)/60.
      IF(ABS(A).GT.89.9)A=SIGN(89.9,A)
      A=D2R*A
      IF(MNS(II).EQ.'S')A=-A
      BB=D2R*(LOD(II)+LOM(II)/60.)
      IF(MEW(II).EQ.'W')BB=-BB
      T1=COS(BB)*COS(A)
      T2=SIN(BB)*COS(A)
      T3=SIN(A)
      TDC=T1*CMT1+T2*CMT2+T3*CMT3
      IF(ABS(TDC).GE..99999)TDC=SIGN(.99999,TDC)
      DEMON=COS(A)*SQRT(1-TDC**2)
      SINB=(CMT2*T1-CMT1*T2)
      COSB=(CMT3-T3*TDC)/DEMON
      BBB=ARCCOS(COSB)
      BBB=R2D*BBB
      IF(SINB.LT.0)BBB=360-BBB
      BSUM(NCASE,I)=BBB
      APA=BBB-ADEG(II)
      APA=ABS(APA)
      IF(APA.GT.90)APA=180-APA
      TZ=D2R*APA
      CSUM(NCASE,I)=1/SQRT((COS(TZ)/AXES2(II,1))**2+(SIN(TZ)/AXES2(I
     *I,2))**2)
      TDC=T1*CMT1+T2*CMT2+T3*CMT3
      DSUM(NCASE,I)=3444.*ARCCOS(TDC)
      ESUM(NCASE,I)=AA2(II)
      FSUM(NCASE,I)=VAL(II)
      LSUM(NCASE,I)=0
      HSUM(NCASE,I)=0.
      DO 218 J=1,N
      IF(INS(II,J).EQ.0)GO TO 218
      LSUM(NCASE,I)=LSUM(NCASE,I)+1
      HSUM(NCASE,I)=HSUM(NCASE,I)+(SIN(DIST(II,J)/3444.)*TAN(ERR(II,J)/5
     +7.29575)*3444.)**2
  218 CONTINUE
      WRITE(LW,7018)
 7018 FORMAT(/)
      WRITE(LW,7011)((LAD(INN(I)),LAM(INN(I)),LAS(INN(I)),MNS(INN(I)),
     1LOD(INN(I)),LOM(INN(I)),LOS(INN(I)),MEW(INN(I))),I=1,NPRT)
 7011 FORMAT(' BPE:',12X,3(I2,'-',I2,'-',I2,A1,5X,I3,'-',I2,'-',I2,A1,
     110X,))
      WRITE(LW,7014)(AA2(INN(I)),I=1,NPRT)
```

```
 7014 FORMAT(/' AREA:'16X,3(F12.0,24X))
      WRITE(LW,7015)(((AXES2(INN(I),J),J=1,2),ADEG(INN(I))),I=1,NPRT)
 7015 FORMAT(' AXES:',10X,3(F7.1,' X ',F7.1,' (',F4.0,')',10X))
      WRITE(LW,7016)(IHEADR,I=1,NPRT)
 7016 FORMAT(//3X,'BEARINGS',8X,3(4A6,10X))
      DO 220 I=1,N
      WRITE(LW,7017)IDD(I),BRNG(I),((INS(INN(J),I),DISI(INN(J),I)
     1,ERR(INN(J),I)),J=1,NPRT)
 7017 FORMAT(2X,A2,F5.1,8X,3(I1,F11.1,4X,F6.1,12X))
  220 CONTINUE
      IF(KWCMT.NE.1)GO TO 7
      WRITE(LW,7032)
      DO 230 J=1,NPRT
      II=FSUM(NCASE,J)
      TP=DSUM(NCASE,J)/CSUM(NCASE,J)
      HSLS=HSUM(NCASE,J)/LSUM(NCASE,J)
      WRITE(LW,7031)BSUM(NCASE,J),CSUM(NCASE,J)
     *,DSUM(NCASE,J),TP,ESUM(NCASE,J),
     *LSUM(NCASE,J),II,HSLS
  230 CONTINUE
      GO TO 7
  250 IF(KWCMT.NE.1)GO TO 280
      WRITE(LW,7075)
      DO 275 I=1,NCASE
      WRITE(LW,7030)(ITSUM(I,J),J=1,2),NMSUM(I)
      WRITE(LW,7032)
 7030 FORMAT(//2X,2A5,3X,I4)
      IF(NMSUM(I).EQ.0)GO TO 275
      DO 260 J=1,NMSUM(I)
      II=FSUM(I,J)
      TP=DSUM(I,J)/CSUM(I,J)
      HSLS=HSUM(I,J)/LSUM(I,J)
      WRITE(LW,7031) BSUM(I,J),CSUM(I,J),DSUM(I,J),
     *TP,ESUM(I,J),LSUM(I,J),II,HSLS
 7031 FORMAT(/1X,F6.1,F12.1,F11.1,F10.3,F10.0,I9,I13,F19.3)
 7032 FORMAT(/////'  BEARING      DIST TO     DIST TO     RATIO      SEARCH
     * INCLUDED   FORWARD     COCKED HAT',/,'    TO CMT       ELLIPSE
     * CMT      DC/DE       AREA        BEARINGS    BEARINGS     SIZE')
  260 CONTINUE
  275 CONTINUE
  280 STOP
      END
```

```
      FUNCTION ARCCOS(A)
C++ THIS FUNCTION CALCULATES THE ARCCOS OF A.  IT ALLOWS THE
C++ ARGUMENT TO BE GREATER THAN ONE SINCE ROUND OFF ERRORS CAN
C++ GIVE SLIGHTLY WRONG VALUES OF A.
      ARCCOS=0.
      IF(ABS(A).LT.1)ARCCOS=ARCOS(A)
      RETURN
      END
```

```
      FUNCTION ARCSIN(A)
C++ THIS FUNCTION CALCULATES THE ARCSIN OF A.  IT ALLOWS THE
C++ ARGUMENT TO BE GREATER THAN ONE SINCE ROUND OFF ERRORS CAN
C++ GIVE SLIGHTLY WRONG VALUES OF A.
      ARCSIN=SIGN(1.570796327,A)
      IF(ABS(A).LT.1)ARCSIN=ARSIN(A)
      RETURN
      END
```

```
      SUBROUTINE ASORT(X,N,IORD)
C++ THIS ROUTINE SORTS THE ARRAY X AND RETURNS THE INDEX OF
C++ THE LARGEST IN IORD(I),THE SECOND LARGEST IN IORD(2), ETC.
      DIMENSION X(1),IORD(I),TEMP(25)
      DO 10 I=1,N
      IORD(I)=I
   10 TEMP(I)=X(I)
      IF(N.LE.1)RETURN
      NL1=N-1
   50 DO 60 I=1,NL1
      IP1=I+1
      DO 60 J=IP1,N
      IF(TEMP(I).GE.TEMP(J))GO TO 60
      ISAV=IORD(J)
      IORD(J)=IORD(I)
      IORD(I)=ISAV
      SAV=TEMP(J)
```

```
      TEMP(J)=TEMP(I)
      TEMP(I)=SAV
   60 CONTINUE
  100 RETURN
      END
```

```
      FUNCTION AVAL(N,MDS,DIST,ERR,INN,JREV)
C++THIS SUBROUTINE CALCULATES A VALUE TO DETERMINE THE "BEST" MODE.
C++THE VALUE USED IS THE NUMBER OF FORWARD BEARINGS PLUS 1/HAT SIZE.
C++[TR 79-5,22].
      DIMENSION JREV(1),INN(1),DIST(20,20),ERR(20,20)
      I1=0
      I2=0
      HATSIZ=0.
      DO 100 I=1,N
      IF(INN(I).NE.1)GO TO 100
C++ HATSIZ IS THE MEAN PERPENDICULAR DISTANCE FROM THE BPE
C++ TO THE INCLUDED BEARINGS.
      HATSIZ=HATSIZ+(SIN(DIST(MDS,I)/3444.)*TAN(ERR(MDS,
     &I)/57.29578)*3444.)**2
      I1=I1+1
      IF(JREV(I).EQ.1)GO TO 100
      I2=I2+1
  100 CONTINUE
C++HATSIZ MUST BE LIMITED TO A NUMBER GREATER THAT 1 SO THAT IT DOES
C++OVERRIDE THE NUMBER OF FORWARD BEARING AS THE PRIMARY CRITERION.
C++ THE HATSIZ IS FORCED TO BE SLIGHTLY GREATER THAN 1 SO THAT IT
C++ WILL NOT BECOME A PRIMARY CRITERION FOR ORDERING THE MODES.
      HATSIZ=AMAX1(1.00001,HATSIZ/I1)
      AVAL=I2+1./HATSIZ
      RETURN
      END
```

```
      SUBROUTINE CFIT(N,INC,NBRG,NINC)
C++ UPDATE INFORMATION IN COMMON FOR THE NEW BPE.
C++ N-NUMBER OF BEARINGS TO BE UPDATED. INC-INDEX ARRAY OF
C++ SPECIFIC BEARINGS TO BE USED.  NBRG-NUMBER OF BEARINGS
C++ USED AFTER THE UPDATE.  SET NINC TO ZERO TO JUST UPDATE THE
C++ VECTORS AND NOT CHANGE THE INCLUDED BEARING SET.
      COMMON/TTR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FIT(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      DIMENSION INC(20)
      IF(NINC.NE.0)NBRG=0
      DO 10 II=1,N
      I=INC(II)
      IF(NINC.NE.0)INN(I)=1
C++ CALCULATE S DOT T,B DOT T,AND 1-(S DOT T)**2
      DST(I)=DOT(S(1,I),T)
      DBT(I)=DOT(B(1,I),T)
      D(I)=1-DST(I)**2
C++ GET THE NEW STANDARD DEVIATION
      IF(NINC.NE.0)ASIG(I)=SIGC(D(I),SIGSAV(I),DST(I))
      IF(NINC.EQ.0)ASIG(I)=FSIG(D(I),SIGSAV(I),DST(I))
      IF(D(I).NE.0)GO TO 5
      FIT(I)=1.
      GO TO 9
C++ CALCULATE FIT, THE COSINE(ANGULAR ERROR), USING THE EQUATION GIVEN
C++ ON (TR 79-5:17).
    5 FIT(I)=DBT(I)/SQRT(D(I))
C++ IF THE ANGULAR ERROR IS GREATER THAN CC*SIGMA CONSIDER
C++ THE BEARING TO BE ON THE FLAT PART OF THE PROBABILITY
C++ DENSITY FUNCTION.  THIS CONSIDERATION MEANS THAT THE
C++ BEARING WILL NOT AFFECT THE MOVEMENT OF THE BPE.
      IF(NINC.EQ.0)GO TO 10
      IF(FIT(I).LT.COS(CC(I)*ASIG(I)))INN(I)=0
    9 IF(NINC.NE.0)NBRG=NBRG+INN(I)
   10 CONTINUE
      DO 20 I=1,N
C++ NORMALIZE USING THE VARIANCE
   20 FIT(I)=FIT(I)/(ASIG(I)*ASIG(I))
      RETURN
      END
```

```
      SUBROUTINE CONF2(N,CSTAR,DSTAR,ANGLE,PHI,PC)
C++ CALCULATE THE SEARCH REGION.
C++ THE CONFIDENCE REGION IS CALCULATED UNDER THE ASSUMPTION
C++ THAT (ERROR)/(SIN(D)*SIGMA) IS APPROXIMATELY NORMALLY
C++ DISTRIBUTED WITH MEAN 0 AND VARIANCE SIGO**2.('D' IS THE
C++ DISTANCE FROM THE STATION TO THE BPE, AND SIGMA IS THE
C++ RATIO OF THE TRUE STANDARD DEVIATION TO SIGO).  SIGO
C++ IS ASSUMED KNOWN AND EQUAL TO 1 (TR 79-5:30).
      COMMON/TTR/OST(20),DBT(20),B(3,20),S(3,20),T(3),FIT(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      COMMON/SAVE/LAD(20),LAM(20),CAS(20),COD(20),LOM(20),LOS(20),
     +MNS(20),MEW(20),AA2(20),AXES2(20,2),VAL(20),
     +NABT(20),INS(20,20),DIST(20,20),ERR(20,20),IDO(20),BRNG(20),ADEG(2
     +0),ITITLE(12)
      A=0
      BB=0
      C=0
      NBRG=0
      DO 100 I=1,N
      IF(INN(I).EQ.0)GO TO 100
      DD=D(I)*D(I)*(COS(PHI)**2)*(ASIG(I)*ASIG(I))
      A=A+((S(3,I)-OST(I)*T(3))**2)/DD
      BB=BB+((S(2,I)*T(1)-S(1,I)*T(2))*(S(3,I)-OST(I)*T(3)))/DD
      C=C+((S(2,I)*T(1)-S(1,I)*T(2))**2)/DD
      NBRG=NBRG+1
  100 CONTINUE
      A=-A/2
      C=-C/2
      ALN=ALOG(1-PC)
C++ AALN IS APPROXIMATELY THE CRITICAL F VALUE FOR 2 AND
C++ INFINITE DEGREES OF FREEDOM WHICH IS USED IF THE VARIANCES
C++ ARE ASSUMED KNOWN.
      IF((A-C).NE.0)GO TO 200
      ANGLE=.7853982
      GO TO 300
  200 ANGLE=ATAN(BB/(A-C))/2
  300 CA=COS(ANGLE)
      SA=SIN(ANGLE)
      CSTAR=SQRT(ABS(ALN/(A*CA*CA+BB*CA*SA+C*SA*SA)))*3444.
      DSTAR=SQRT(ABS(ALN/(A*SA*SA-BB*SA*CA+C*CA*CA)))*3444.
      IF(CSTAR.GE.DSTAR)ANGLE=1.5707963-ANGLE
      IF(CSTAR.LT.DSTAR)ANGLE=-ANGLE
      IF(ANGLE.LT.0)ANGLE=ANGLE+1.5707963
```

```
      CSTAR=AMAX1(CSTAR,1.)
      DSTAR=AMAX1(DSTAR,1.)
      RETURN
      END


      SUBROUTINE CROSS(A,B,C)
C++ CALCULATE THE CROSS PRODUCT A CROSS B AND RETURN THE
C++ RESULT IN C
C++ THIS SUBROUTINE IS CALLED USING TWO DIMENSIONAL ARRAYS WITH THE
C++ FIRST INDEX REPRESENTING THE VECTOR COMPONENTS.
      DIMENSION A(3),B(3),C(3)
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
      RETURN
      END


      SUBROUTINE DMS(AA,AD,AM,AS,IH,IOPT)
C++ CONVERTS A DECIMAL ANGLE TO DEGREES,MINUTES AND SECONDS.
      INTEGER AD,AM,AS
      DATA NN,NS,NE,NW/'N','S','E','W'/
      A=AA*57.29577951
      IF(IOPT.EQ.1)GO TO 10
      IH=NN
      IF(A.LT.0)IH=NS
      GO TO 20
   10 IH=NE
      IF(A.LT.0)IH=NW
   20 I=ABS(A)
      J=60*(ABS(A)-I)
      AJ=60.*(ABS(A)-I)
      K=60*(AJ-J)
      AD=I
      AM=J
      AS=K
```

```
      RETURN
      END


      FUNCTION DOT(A,C)
C++ CALCULATE THE DOT PRODUCT OF TWO VECTORS.
C++ THIS FUNCTION IS CALLED USING TWO DIMENSIONAL ARRAYS WITH THE
C++ FIRST INDEX REPRESENTING THE VECTOR COMPONENTS.
      DIMENSION A(3),C(3)
      DOT=A(1)*C(1)+A(2)*C(2)+A(3)*C(3)
      RETURN
      END


      SUBROUTINE FLAGCK(N,IN,JN)
C++ FIND A BEARING INTERSECTION WHICH HAS NOT BEEN USED.
      COMMON/FLAG/IFLAG(20,20),ISTRT,JSTRT
      NL1=N-1
      IF(ISTRT.GT.NL1)GO TO 12
      DO 10 I=ISTRT,NL1
      IP1=I+1
      IF(I.EQ.ISTRT)IP1=JSTRT
      DO 10 J=IP1,N
      IF(IFLAG(I,J).EQ.1)GO TO 10
      IN=I
      JN=J
      ISTRT=I
      JSTRT=J+1
      IF(J.LT.N)RETURN
      ISTRT=I+1
      JSTRT=I+2
      RETURN
   10 CONTINUE
   12 IN=0
      JN=0
      RETURN
      END
```

```
      SUBROUTINE FLAGST(NN,INC)
C++ SET THE IFLAG ARRAY VALUES TO 1 (INDICATING THE INTERSECTION
C++ CANNOT BE USED AS A STARTING POINT) FOR ALL INTERSECTIONS OF
C++ BEARINGS WHICH HAVE BOTH BEEN USED IN THE CURRENT FIX.
      COMMON/FLAG/IFLAG(20,20),ISTRT,JSTRT
      DIMENSION INC(20)
      NL1=NN-1
      DO 10 I=1,NL1
      IP1=I+1
      DO 10 J=IP1,NN
      IN=INC(I)*INC(J)
      IF(IN.EQ.1)IFLAG(I,J)=IN
   10 CONTINUE
      RETURN
      END
```

```
      FUNCTION FSIG(SQ,SIGM,SDT)
C++   THIS FUNCTION CALCULATES THE STANDARD DEVIATIONS OF A BEARING AS
C++   A FUNCTION OF THE RANGE FROM THE STATION TO THE TARGET VECTOR.
C++   THE PROCEDURE IS USED TO CALCULATE THE FINAL STANDARD DEVIATION
C++   FOR INPUT TO THE REFIX PROCEDURE.  IT IS DIFFERENT FROM SIGC IN
C++   THAT IT ELIMINATES THE KEYHOLE AROUND THE STATION.(TR 79-5;23,26)
C++
C++   SQ IS THE SINE**2 OF THE DISTANCE FROM THE STATION TO THE TARGET.
C++   SDT IS THE COSINE OF THE DISTANCE.  SIGM IS THE INPUT STANDARD
C++   DEVIATION AND IS USED AS THE LONG RANGE STANDARD DEVIATION.
      F=SIGM/1.826352
C++  THE CONSTANT 1.826352 IS THE LONG RANGE MULTIPLIER OF THE STANDARD
C++  DEVIATION DERIVED FROM THE ROSS RANGE CURVE.
      S=SQRT(SQ)
      DIST=ARCSIN(S)
      IF(SDT.LT.0)DIST=3.141593-DIST
      DIST=DIST*3444.
      IF(DIST.LE.600.)FSIG=.75+DIST/2400.
      IF(DIST.GT.600..AND.DIST.LE.5410.)FSIG=.826352+S
      IF(DIST.GT.5410.)FSIG=1.826352
   20 FSIG=FSIG*F
      RETURN
      END
```

```
      FUNCTION PHINEW(A,N)
C++ CALCULATE A NEW LATITUDE.(TR 79-5:19)
      COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FII(20),D(20)
     *,CC(20),IPEV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      S1=0.
      S2=0.
      DO 10 I=1,N
      IF(D(I).EQ.0.OR.DBT(I).EQ.0)GO TO 10
      S1=S1+INN(I)*FII(I)*(B(3,I)/DBT(I)+S(3,I)*DSI(I)/D(I))
      S2=S2+INN(I)*FII(I)*((B(1,I)*COS(A)+B(2,I)*SIN(A))/DBT(I)+DST(I)*
     *(S(1,I)*COS(A)+S(2,I)*SIN(A))/D(I))
   10 CONTINUE
      IF(S2.EQ.0)GO TO 20
      PHINEW=ATAN(S1/S2)
      RETURN
   20 PHINEW=1.570796327
      IF(S1.LT.0)PHINEW=-PHINEW
      RETURN
      END


      SUBROUTINE REFIX(THETA,PHI,NBRG,BRNG)
C++SUBROUTINE REFIX IS A PROCEDURE USING A STANDARD REGRESSION
C++BEARING SET.  THETA AND PHI ARE INITIAL ESTIMATES OF THE
C++BPE AND ARE USED AS THE ORIGIN FOR THE REGRESSION PROBLEM.
C++(TR 79-5:27+
      COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FII(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      DIMENSION BRNG(1)
      D2R=.017453293
C++THE FOLLOWING ARE COMPONENTS OF THE X-TRANSPOSE-X AND X-TRANSPOSE-Y
C++MATRICES.
      XTX11=0.
      XTX12=0.
      XTX22=0.
      XTY11=0.
      XTY21=0.
      DO 100 I=1,NBRG
      SIGMA=FSIG(D(I),SIGSAV(I),DST(I))
      ASIG(I)=SIGMA
```

```
      IF(IWN(I).NE.1)GO TO 100
      R=SQRT(1-DST(I)**2)
      DD=R*COS(PHI)
      COSB=(S(3,I)-DST(I)*T(3))/DD
      SINB=(S(2,I)*T(1)-S(1,I)*T(2))/DD
      COSA=(T(3)-DST(I)*S(3,I))/(R*SQRT(1-S(3,I)**2))
C++BETA IS THE BEARING FROM THE ORIGIN TO THE STATION.  ALPHA IS THE
C++BEARING FROM THE STATION TO THE ORIGIN.
      BETA=ARCCOS(COSB)/D2R
      IF(SINB.LT.0)BETA=360-BETA
      ALPHA=ARCCOS(COSA)/D2R
      IF(T(2)*S(1,I)-T(1)*S(2,I).LT.0)ALPHA=360-ALPHA
      BB=BNZ(I)
      ERR=BB-ALPHA
      IF(ABS(ERR).GT.180.)ERR=-SIGN(360.-ABS(ERR),ERR)
      IF(ABS(ERR).LT.90.)GO TO 50
      BB=AMOD(BB+180.,360.)
      ERR=BB-ALPHA
      IF(ABS(ERR).GT.180.)ERR=-SIGN(360.-ABS(ERR),ERR)
   50 ERR=ERR*D2R
C++DELTA IS THE PERPENDICULAR DISTANCE FROM THE ORIGIN TO THE BEARING.
      DELTA=ERR*R*3444.
      SS=SIGMA*R*3444.
      SSS=SS*SS
      IF(R.GT..087.OR.DST(I).LT.0.)GO TO 60
      COSB=COS((BB+180.)*D2R)
      SINB=SIN((BB+180.)*D2R)
   60 XTX11=XTX11+(COSB*COSB)/SSS
      XTX12=XTX12-(SINB*COSB)/SSS
      XTX22=XTX22+(SINB*SINB)/SSS
      XTY11=XTY11-(COSB*DELTA)/SSS
      XTY21=XTY21+(SINB*DELTA)/SSS
  100 CONTINUE
      DET=XTX11*XTX22-XTX12*XTX12
      IF(DET.GT..0000001)GO TO 150
      RETURN
  150 XHAT=(XTX22*XTY11-XTX12*XTY21)/DET
      YHAT=(XTX11*XTY21-XTX12*XTY11)/DET
      T(1)=T(1)-(SIN(PHI)*COS(THETA)*YHAT+SIN(THETA)*XHAT)/3444.
      T(2)=T(2)+(COS(THETA)*XHAT-SIN(PHI)*SIN(THETA)*YHAT)/3444.
      T(3)=T(3)+COS(PHI)*YHAT/3444.
      DET=SQRT(T(1)*T(1)+T(2)*T(2)+T(3)*T(3))
      T(1)=T(1)/DET
```

```
      T(2)=T(2)/DET
      T(3)=T(3)/DET
      PHI=ARSIN(T(3))
      THETA=ARCCOS(T(1)/COS(PHI))
      IF(T(2).LT.0)THETA=-THETA
      RETURN
      END


      SUBROUTINE REVBRG(N)
C++ REVERSE BEARING.  IREV INDICATES WHICH BEARINGS HAVE BEEN
C++ REVERSED.
      COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FIT(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      DO 200 I=1,N
      DBT(I)=DOT(B(1,I),T)
      IF(DBT(I).GE.0)GO TO 200
      DO 94 JJ=1,3
   94 B(JJ,I)=-B(JJ,I)
      JREV(I)=1
      IF(DOT(S(1,I),T).LT.0)IREV(I)=1
  200 CONTINUE
      RETURN
      END


      FUNCTION SIGC(SD,SIGM,SDT)
C++   THIS FUNCTION CALCULATES THE STANDARD DEVIATIONS OF A BEARING AS
C++   A FUNCTION OF THE RANGE FROM THE STATION TO THE TARGET VECTOR.
C++   THE PROCEDURE USES AN APPROXIMATION TO THE RUSS RANGE CURVE.
C++   (TR 79-5:23)
C++
C++   SD IS THE SINE**2 OF THE DISTANCE FROM THE STATION TO THE TARGET.
C++   SDT IS THE COSINE OF THE DISTANCE.  SIGM IS THE INPUT STANDARD
C++   DEVIATION AND IS USED AS THE LONG RANGE STANDARD DEVIATION.
      F=SIGM/1.826352
C++  THE CONSTANT 1.826352 IS THE LONG RANGE MULTIPLIER OF THE STANDARD
```

```
C++  DEVIATION DERIVED FROM THE ROSS RANGE CURVE.
      S=SQRT(SQ)
      SIN01=.0174524
      SIN02=.034899
      SIN10=.173648
C++  IF SQT IS LESS THAN 0, THE DISTANCE IS GREATER THAN 90 DEGREES.
      IF(SQT.LT.0)GO TO 10
      SIGC=60
C++  THE CONSTANTS BELOW ARE DERIVED FROM THE ROSS RANGE CURVE.  FOR
C++  CLOSE IN DISTANCES, THE VARIANCE HAS BEEN INCREASED TO PREVENT
C++  UNWARRENTED ELIMINATION OF BEARINGS FROM NEAR STATIONS.
      IF(S.GT.SIN01.AND.S.LE.SIN02)SIGC=60.-3152.387*(S-.0174524)
      IF(S.GT.SIN02.AND.S.LE.SIN10)SIGC=.174524/S
      IF(S.GT.SIN10)SIGC=S+.826352
      GO TO 20
   10 SIGC=1.826352
   20 SIGC=SIGC*F
      RETURN
      END


      SUBROUTINE STRTPT(N,PHI,THETA,KC,IND)
C++ FIND A STARTING POINT.
      COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FII(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      COMMON/STAT/NST,ID(27),ALA(27),ALO(27),CCC(27),SIG(27)
      COMMON/FLAG/IFLAG(20,20),ISTRT,JSTRT
      COMMON/SAVE/LAD(20),LAM(20),LAS(20),LOD(20),LOM(20),LOS(20),
     +MNS(20),MEW(20),AA2(20),AXES2(20,2),VAL(20),
     +NABT(20),INS(20,20),DIST(20,20),ERR(20,20),IUD(20),BRNG(20),ADEG(2
     +0),ITITLE(12)
      DIMENSION BRNGG(20),TT(3),TS(3)
      R2D=57.29577951
      D2R=.01745329
C+++++++++ WRITE CONTROL ++++++++++++++++++++++++++
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARDS$
      KW=0
      LW=7
      SIN20=.342020
    1 CALL FLAGCK(N,IN,JN)
```

```
      IF(KW.EQ.1)WRITE(LW,7000)IN,IDD(IN),JN,IDD(JN)
 7000 FORMAT(/1'  INTERSECTION OF BEARINGS ',I3,' (',A2,') AND ',I3,' (',
     +A2,') IS BEING TRIED AS A STARTING POINT.')
      IF(IN.NE.0)GO TO 5
      IND=0
C++IND IS SET TO 0 WHEN ALL STARTING POINTS HAVE BEEN USED UP.
      RETURN
    5 IF(KC.NE.0)GO TO 10
      IFLAG(IN,JN)=1
      IFLAG(JN,IN)=1
   10 CALL CROSS(S(1,IN),B(1,IN),TT)
      CALL CROSS(S(1,JN),B(1,JN),TS)
      CALL CROSS(TT,TS,T)
C++ THE VECTOR T POINTS TOWARD ONE INTERSECTION OF THE BEARING
C++ PLANES.  IF D1 AND D2 ARE BOTH POSITIVE, BOTH BEARINGS ARE
C++ POINTING TOWARD THE INTERSECTION--USE IT AS A STARTING
C++ POINT. IF D1 AND D2 ARE BOTH NEGATIVE, BOTH BEARINGS ARE
C++ POINTING AWAY FROM THE INTERSECTION--USE THE ANTIPUDAL
C++ INTERSECTION AS THE STARTING POINT.
      D1=DOT(B(1,IN),T)
      D2=DOT(B(1,JN),T)
      AMG=SQRT(T(1)*T(1)+T(2)*T(2)+T(3)*T(3))
      DO 20 I=1,3
   20 T(I)=T(I)/AMG
      IF(D1.GE.0.AND.D2.GE.0)GO TO 40
      IF(D1.LE.0.AND.D2.LE.0)GO TO 37
      D1=DOT(S(1,IN),T)
      D2=DOT(S(1,JN),T)
      IF(D1.GE..9848.OR.D2.GE..9848)GO TO 40
      IF(D1.LE.(-.9848).OR.D2.LE.(-.9848))GO TO 37
      IF(KW.EQ.1)WRITE(LW,7002)IN,JN
 7002 FORMAT('  INTERSECTION: ',2I3,' ELIMINATED BECAUSE OF REVERSED BEAR
     +ING.')
      GO TO 1
   37 DO 38 I=1,3
   38 T(I)=-T(I)
C ++  IF THE INTERSECTION ANGLE IS TO SMALL DO NOT USE THE POINT.
   40 IF(AMG.GT.KC*SIN20)GO TO 41
      IF(KW.EQ.1)WRITE(LW,7001)IN,JN
 7001 FORMAT('  INTERSECTION: ',2I3,' ELIMINATED BECAUSE INTERSECTION ANG
     +LE TOO SMALL.')
      GO TO 1
   41 PHI=ARCSIN(T(3))
```

```
      COP=COS(PHI)
      THETA=ATAN2(T(2)/COP,T(1)/COP)
      IFLAG(IN,JN)=1
      IFLAG(JN,IN)=1
      RETURN
      END
```

```
      FUNCTION THETAN(A,N)
C++ CALCULATES A NEW LONGITUDE.(TR 79-5:19)
      COMMON/ITR/DST(20),DBT(20),B(3,20),S(3,20),T(3),FIT(20),D(20)
     *,CC(20),IREV(20),JREV(20),BSAV(3,20),SIGSAV(20),ASIG(20),INN(20)
      AA=180.*.017453293
      BB=45.*.017453293
      S1=0.
      S2=0.
      DO 10 I=1,N
      IF(D(I).EQ.0.OR.DBT(I).EQ.0)GO TO 10
      S1=S1+INN(I)*FIT(I)*(B(2,I)/DBT(I)+S(2,I)*DSI(I)/D(I))
      S2=S2+INN(I)*FIT(I)*(B(1,I)/DBT(I)+S(1,I)*DSI(I)/D(I))
   10 CONTINUE
      IF(S2.EQ.0)GO TO 20
C++ PUT THE NEW THETA IN THE CORRECT QUADRANT.
      THETAN=ATAN(S1/S2)
      THET=THETAN
      IF(THETAN.GE.0.AND.(A.LT.-BB.OR.A.GT.3*BB))THET=THETAN-AA
      IF(THETAN.LE.0.AND.(A.GT.BB.OR.A.LT.-3*BB))THET=THETAN+AA
      THETAN=THET
      RETURN
   20 THETAN=1.570796327
      IF(A.LT.0)THETAN=-THATAN
      RETURN
      END
```

```
      BLOCK DATA
      COMMON/STAT/NST,ID(27),ALA(27),ALO(27),CCC(27),SIG(27)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$OPTION CARD$$
C++THE ARRAY CCC GIVES THE NUMBER OF STANDARD DEVIATIONS FOR BEARING IN-
C++CLUSION FOR EACH STATION.  THE DATA STATEMENT MUST REFLECT THE ORDER
C++OF THE STATIONS IN THE STATION INPUT FILE.
      DATA CCC/3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,
     +3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0/
      END
```

Appendix C

Sample Output

TEST CASE NR 1

NUMBER OF BEARINGS: 6 CMT LALO: 39- 0- ON 105- 0- OW NUMBER OF NODES: 1

BPE: 39-29-29N 104-57- OW

AREA: 3300.
AXES: 34.6 X 30.4 (53.)

BEARINGS		INC	DISTANCE	ERROR
PB	212.7	1	651.7	0.1
WR	105.8	1	913.7	-0.3
SR	61.7	1	799.5	0.5
BR	335.5	1	893.3	-0.5
MI	311.1	1	1486.7	1.4
NF	285.5	1	1364.5	-0.4

BEARING TO CMT	DIST TO ELLIPSE	DIST TO CMT	RATIO DC/DE	SEARCH AREA	INCLUDED BEARINGS	FORWARD BEARINGS	COCKED HAT SIZE
184.6	32.0	29.1	0.911	3300.	6	6	239.405

TEST CASE NR 2

NUMBER OF BEARINGS: 7 CMT LALO: 38-18- 0N 85-54- 0W NUMBER OF MODES: 2

BPE:	39-27-44N	104-56-29W		38-29-23N	86- 2-13W		
AREA:		4548.			2789.		
AXES:	47.0 X	30.8 (55.)		41.0 X	21.6 (92.)		

BEARINGS		INC	DISTANCE	ERROR	INC	DISTANCE	ERROR
PB	212.7	1	653.1	0.2	0	795.3	75.1
WR	105.8	1	914.9	-0.4	0	1730.1	15.5
SR	61.7	1	799.3	0.3	0	1663.5	-9.0
BR	34.5	0	891.5	57.5	1	948.7	-0.4
MI	341.1	0	1485.5	31.4	1	816.3	0.6
NF	285.5	1	1364.1	-0.3	1	480.5	-0.4
LI	261.8	0	1505.5	-15.6	1	665.3	0.3

BEARING TO CMT	DIST TO ELLIPSE	DIST TO CMT	RATIO DC/DE	SEARCH AREA	INCLUDED BEARINGS	FORWARD BEARINGS	C CKED MAT SIZE
88.4	39.7	891.7	22.476	4548.	4	4	28.985
135.6	37.5	12.7	0.462	2789.	4	4	32.191

TEST CASE NR 3

NUMBER OF BEARINGS: 5 CMT LALO: 36-50- 0N 75-48- 0W NUMBER OF MODES: 2

BPE:	36-46-17N	75-56-59W			36-22- 5N	77-20-17W	
AREA:		159.				238.	
AXES:	50.5 X	1.0 (70.)			28.8 X	2.6 (69.)	

BEARINGS		INC	DISTANCE	ERROR	INC	DISTANCE	ERROR
PI	211.0	1	715.3	-0.6	0	768.0	-4.3
LI	224.6	0	314.2	7.3	1	374.3	0.9
NF	71.3	1	17.8	1.0	1	58.5	-179.1
MI	13.0	0	696.1	-4.6	1	653.2	0.2
PB	120.8	1	1186.6	0.9	1	1157.4	-2.3

BEARING TO CMT	DIST TO ELLIPSE	DIST TO CMT	RATIO DC/DE	SEARCH AREA	INCLUDED BEARINGS	FORWARD BEARINGS	COCKED HAT SIZE
74.8	12.6	7.6	0.598	159.	3	3	134.665
68.8	28.8	79.1	2.752	238.	4	3	333.164

TEST CASE NR 4

NUMBER OF BEARINGS: 8 CMT LALO: 39-20- 0N 85-50- 0W NUMBER OF NODES: 5

BPE:	38-29-23N	86- 2-13W		39-26-38N	104-38-24W		41-39-51N	93- 3- 5W	
AREA:		2789.			11285.			10085.	
AXES:	41.0 X	21.6 (92.)		94.9 X	37.8 (90.)		73.7 X	43.[illegible] (125.)	

BEARINGS		INC	DISTANCE	ERROR	INC	DISTANCE	ERROR	INC	DISTANCE	ERROR
WR	105.8	0	1730.1	15.5	1	927.7	-0.0	0	1779.1	23.4
SR	61.7	0	1663.5	9.0	1	812.1	-0.1	1	1803.2	-2.5
BR	34.5	1	948.7	-0.4	0	884.3	56.7	1	1182.6	0.9
MI	341.1	1	816.3	8.6	0	1472.5	31.2	0	963.3	-11.4
NF	285.4	1	480.5	-8.4	1	1350.2	-0.2	0	436.7	-29.7
LI	261.8	1	665.3	8.3	0	1492.2	-15.4	0	500.1	-16.4
PI	211.0	0	901.0	-31.7	0	1643.6	-56.0	0	717.5	-36.8
PB	120.8	0	795.3	-16.8	0	647.8	-90.6	1	744.6	0.5

BEARING TO CMT	DIST TO ELLIPSE	DIST TO CMT	RATIO DC/DE	SEARCH AREA	INCLUDED BEARINGS	FORWARD BEARINGS	COCKED HAT SIZE
125.7	30.6	13.0	0.426	2789.	4	4	32.191
88.3	94.8	880.8	9.289	11285.	3	3	9.701
213.6	43.8	237.0	5.470	10085.	3	3	1983.536

```
HFDF/STATION (06/18/79)

 100        *
 200   PB  48.9       - 97.4   2.5
 300   VC  49.2       -123.0   2.5
 400   WR  45.4       -124.0   2.5
 500   $R  34.0       -120.1   2.5
 600   BR  26.0       -097.5   2.5
 700   MI  25.8       -080.3   2.5
 800   NF  36.7       -076.2   2.5
 900   LI  41.0       -072.0   2.5
1000   PI  47.2       -068.2   2.5
```

```
HFDF/IN (06/18/79)

 100      TEST CASE NR 1
 200      6 39.00 -105.00
 300     PB 212.7
 400     WR 105.8
 500     $R 061.7
 600     BR 336.5
 700     MI 311.1
 800     NF 285.5
 900      TEST CASE NR 2
1000      7 38.15 -085.54
1100     PB 212.7
1200     WR 105.8
1300     SR 061.7
1400     BR 034.5
1500     MI 341.1
1600     NF 285.5
1700     LI 261.8
1800      TEST CASE NR 3
1900      5 36.50 -075.48
2000     PI 211.0
2100     LI 224.6
2200     NF 071.3
2300     MI 013.0
2400     PB 120.8
2500       TEST CASE NR 4
2600      8 38.20 -085.50
2700     WR 105.9
2800     SR 061.7
2900     BR 034.5
3000     MI 341.1
3100     NF 285.5
3200     LI 261.8
3300     PI 211.0
3400     PB 120.9
```